



## TIME IS MONEY

Get life in order and track what you do



## INSIDE WINE 9.0

Windows apps have never run so fast!



## BOOK SERVER

Stream your ebooks everywhere you go

# LINUX FORMAT

The #1 open source mag



## MASTER CHATGPT

Show the machines who's boss with our AI secrets

# LINUX INVADES WINDOWS

How the plucky Tux rebels infiltrated and took over Windows and Microsoft!

## PLUS: HOW TO

- » Code your own Countdown game
- » Create high-speed collision detection
- » Revive the MK14 – the daddy of the ZX80

**100**  
pages of Linux  
tricks, tips  
& more!

## SMOOTH STARTUPS

Explore the Linux init to gain speed and control

## VALVE PROTON

Fix game glitches on your Steam Deck

## GAME STREAMING

The hacked Nvidia open source gaming system

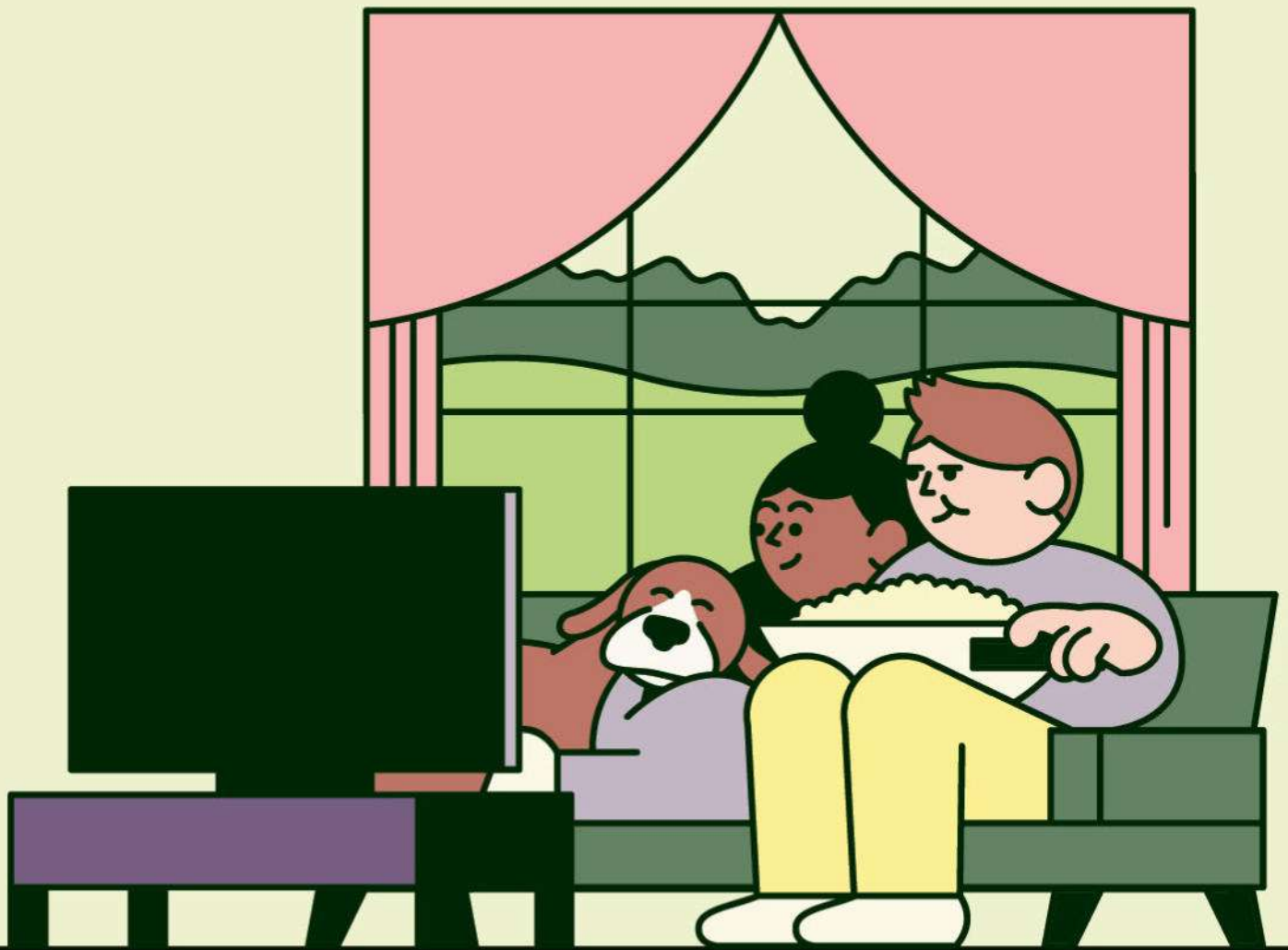
LXF May 2024

FUTURE



# SAVVY SAVERS, STEP THIS WAY.

Discover cover options for car, home and travel insurance, plus broadband and more.



**GO.**  
**COMPARE**

Get more information and compare quotes at [Go.Compare](https://www.Go.Compare).



# LINUX FORMAT



## » MEET THE TEAM

Website trackers put Linux usage at over 4% for the first time. Where do you see Linux use increasing and how do you think things will go in the future?



### Matthew Holder

I think that as devices such as the Steam Deck become more popular, Linux usage will increase without the user even knowing about it. Other areas of growth will be as immutable distros become more and more

popular and provide a more rock-solid, appliance-like usage.



### Nate Drake

*Wine* and *Proton* bridge the gap between Linux and Windows, not to mention the 250 million Windows 10 devices that can't be upgraded unless users switch to Linux. I'm happy to occupy the exclusive

4% niche enjoyed by groups like vegetarians and redheads.



### Nick Peers

It's brilliant news that more people are switching to Linux. I'm guessing these are people with older but perfectly serviceable PCs who don't want to buy a whole new computer just to run Windows 11. Could the trickle become a flood when Windows 10

reaches end of life next year?



### Les Pounder

Valve's Steam Deck has shown that Linux is a viable gaming platform. With SteamOS, *Proton* and underlying technologies to run Windows files on Linux, we have the opportunity to broaden Linux adoption by

showing we don't have to give up our triple AAA games.



### Michael Reed

Linux makes sense as a gaming platform, and that opens many doors. The success of the Steam Deck helps. There are many office scenarios where Linux and a free office suite makes more sense than paying for software.

Windows is changing so much, and some people don't want it.

## Linux is everywhere



Everyone gets excited when the Linux market share goes up. The interwebs were abuzz that desktop Linux hit 4% (see *News*, page 6), never mind the 3% for ChromeOS (which is Linux) and whatever makes up the 6% of "unknown" operating systems – Haiku, is that you? It was more or less this time last year when the same excitement was around Linux hitting 3%, up from 2% the year before. What is going on?

The truth is – and we have talked about this before – Linux is everywhere now, to the point where even Microsoft, 25 years on from its Halloween Documents leak, can't escape and has to embrace Linux and open source. That's why we're looking at how Linux is now embedded inside Windows, how Microsoft has created its own Linux distro, and Linux is the most widely deployed OS on Azure, along with the many open source projects Microsoft contributes to, including the Linux kernel itself. It's a staggering turnaround from a corporation whose past CEO proclaimed Linux to be a cancer.

But that's just further testament to the flexibility and ingenuity of the kernel and the open source community around it. This issue we're covering all areas, from Steam Deck gaming and running Windows tools through *Wine*, serving up ebooks on a shared network system, reliving classic retro computer systems like the MK14 to streaming video for desktop and gaming around your network and getting the most from AI systems. Along with the usual reviews and a time trackers *Roundup*, there's loads to enjoy with open source.

*Neil*

**Neil Mohr** Editor  
neil.mohr@futurenet.com



**Subscribe  
& save!**

On digital and print:  
see page 16



# Contents



## REVIEWS



### AMD Ryzen 5 8600G 19

This reasonably priced processor is the CPU of choice for **Chris Szewczyk** and his tight budgets.



### Deepin 23 20

**Nate Drake** is in too deep with Deepin – is this Chinese-made distribution a crouching tiger or a hidden dragon?



### KaOS 2024.01 21

**Nate Drake** seeks to find some order in KaOS – a distro incorporating the KDE Plasma 6 desktop environment.

### Bluestar Linux 6.7.6 22

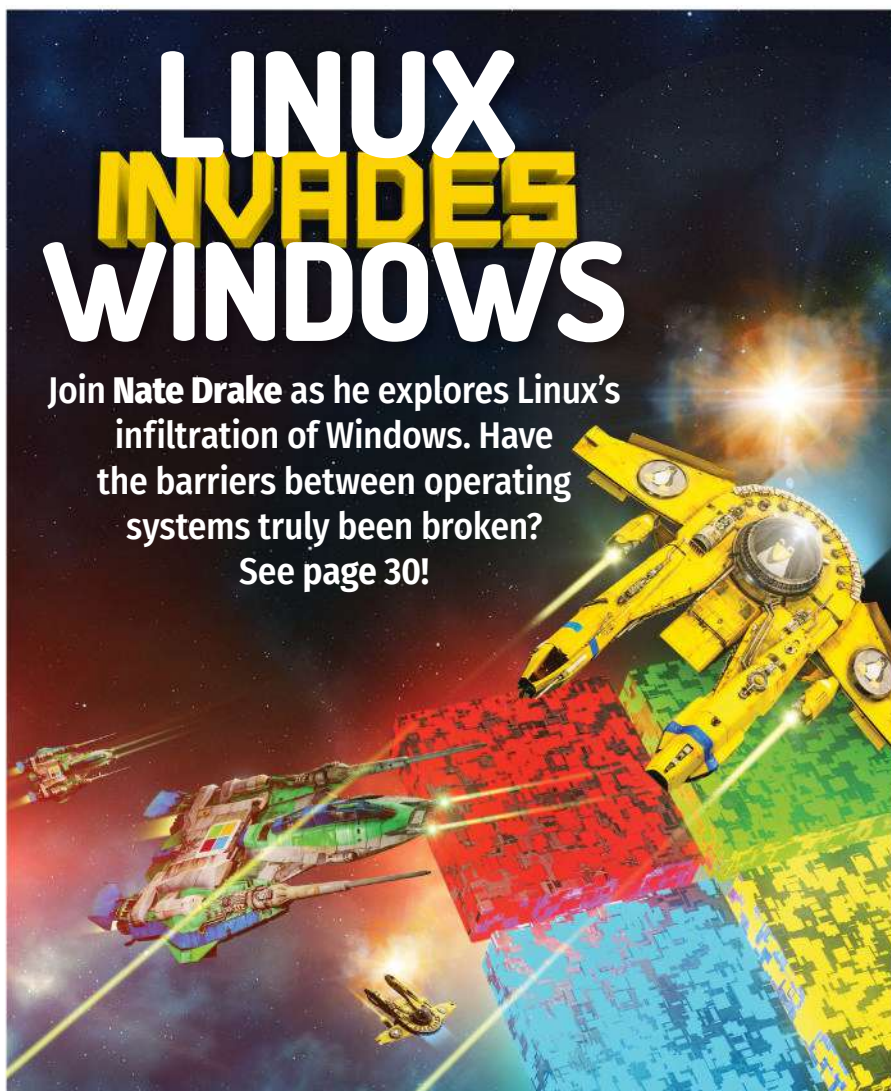
**Nate Drake** finds blue is the colour, after firing up this intuitive and visually stunning Arch-based distro.

### Drauger OS 7.6 23

**Nate Drake** puts this distribution through its paces – is Drauger a gaming dragon or just a plain drag?

## LINUX INVADES WINDOWS

Join **Nate Drake** as he explores Linux's infiltration of Windows. Have the barriers between operating systems truly been broken? See page 30!



CREDIT: Magictorch

## ROUNDUP



### Time trackers 24

This month, **Michael Reed** is keeping a strict eye on himself and trying to improve his work efficiency by investigating time-tracking applications.

## IN DEPTH



### The Grapes of Tux 44

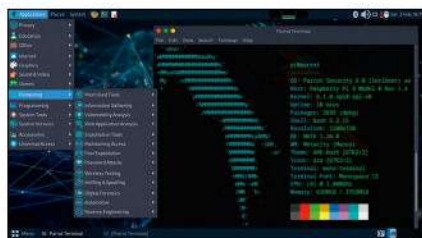
Sommelier **Jonni Bidwell** shows you how to get your pesky Windows applications running on Linux, as he uncorks the latest Wine release.

CREDIT: Getty Images/Stockphoto, AMD

## PI USER



**Pi news** ..... 39  
Introduced by **Les Pounder**, who celebrates the Raspberry Pi's twelfth birthday. Plus updates from the SBC world.



**ParrotOS** ..... 40  
**Les Pounder** has watched *Hackers* at least 12 times and knows all the books. He just needs a cool haircut and a RISC laptop.

## CODING ACADEMY

**Solve word puzzles** ..... 90  
Living conundrum **Matt Holder** discovers how to use Python to improve his puzzle-solving skills when watching *Countdown*.

**Asteroids collision detection in C** ..... 94  
Did those pixels touch or just pass by? **David Bolton** demonstrates how to check collisions with lots of objects on screen.

## REGULARS AT A GLANCE

**News** ..... 6  
This month: Linux love, Plasma 6 premieres, new Nvidia EULA clause, FreeBSD forgoes 32-bit, and the newest Framework laptop ships, plus a round-up of some of the latest distro releases.

**Kernel watch** ..... 10

**Answers** ..... 11  
**Neil Bothwick** learned to count in binary before decimal, which has stood him in good stead when answering queries on mounting MP3 players, deleting invisible files, identifying wrong users, and more.

**Mailserver** ..... 14  
Readers write in with reminiscences and queries, and **Neil Mohr** shares his wisdom.



**Bambu Lab X1-Carbon** ..... 41  
There's a reason this Bambu is shooting up all over and **Andrew Sink** is more than happy to help it out.

**Create smart notifications** ..... 42  
**Les Pounder** loves building gadgets with the £8 Raspberry Pi Pico W – and some of them, like this one, even work.



## TUTORIALS

**TERMINAL: Vulnerability audits** ..... 48  
Invulnerable **Shashank Sharma** has found that *Lynis* is the perfect terminal tool for keeping systems safe.

**KAVITA: Stream your ebook library** ..... 50  
**Nick Peers** looks at a free tool that enables you to collate, read and share your digital publications over the internet.

**SYSTEMD: It's startup time, init!** ..... 54  
**Mats Tage Axelsson** explains how Linux takes you from the kernel to a running system, both today and back in time.



**MOONLIGHT: Home game streaming** ..... 60  
Promising the Sun and the Moon, **Michael Reed**'s heavenly bodies promise perfect gaming streaming on most PC hardware.

**EMULATION: Reviving the MK14** ..... 64  
**Mike Bedford** uses emulation to learn about Sinclair's first sub-£100 computer, which predated the ZX80 by three years.

**OPENCAD: Clever CAD coding** ..... 68  
Hungarian boy genius **Tam Hanna** tidies up his underground bunker's cigar corner with a bit of creative 3D printing.

**PROTON: Fix Steam Deck glitches** ..... 72  
Despite amazing work, there's plenty of incompatible Steam Deck games – but fixes are at hand, though not for **Neil Mohr**.

## ADMINISTERIA

**Administeria** ..... 74  
**Stuart Burns** shares *Docker* container tips and explains why he still loves Ubuntu, despite his criticism of Canonical.

**Collaborate with OnlyOffice** ..... 76  
Nowadays, online document collaboration is a must. **Kseniya Fedoruk** explains how to set up OnlyOffice DocSpace.

**ChatGPT masterclass** ..... 78  
Want to get more from the best-known AI service on the planet? **Barry Collins** and **Neil Mohr** explain how to make AI work for you.





# Newsdesk

**THIS ISSUE:** Linux love » Plasma 6 premieres » New Nvidia EULA clause » FreeBSD forgoes 32-bit » Latest Framework

## OPERATING SYSTEMS

# Linux is popular!

Linux has long been a favoured OS for servers and embedded devices but is now increasingly popular for desktop machines, too.

**A** ccording to data gathered by StatCounter, as of March 2024, Linux accounts for 4.19% of all desktop operating systems. This marks an all-time high for Linux desktop adoption.

Naturally, these figures are nothing compared to the Windows market share during the same period (72%) or even Mac OS (15.46%), but two years ago, Linux desktops only accounted for around 2% of users.

Assuming the Linux install base continues to double every two years, Redditors have pointed out jocularly that we can expect 100% Linux adoption some time in 2033, or even sooner if you count Linux-based ChromeOS.

All joking aside, this is a huge leap considering that it took Linux over 30 years to secure a 3% share of all desktop operating systems, a target it reached in June 2023.

There's most likely a number of factors contributing to more Linux desktop users. We've previously covered how installing the OS is a good alternative to junking non-TPM-2.0-compliant Windows 10 PCs.

Sales of the popular Steam Deck handheld, which can run most Windows games in Linux flawlessly via the *Proton* compatibility layer, might partly account for this spike in usage, although it's unclear how this links through to website browser statistics.

When reporting this story, other tech outlets have emphasised Linux's open source nature making it ideal for technology enthusiasts. They also point out the ready availability of user-friendly distros, such as Ubuntu and Linux Mint. Still, we could counter that it's hardly news that the Linux kernel is open source. Ubuntu has also been around for almost 20 years.

Regardless of the reasons, this raises the perennial question of whether 2024 (or some other arbitrary date in the near future) will be the 'Year of the Linux Desktop'.

By some measures, this has already happened. In 2020, for instance, Chromebook outsold MacBooks two to one. According to StatCounter, in 2024 Android accounts for over 71% of mobile operating systems – though, admittedly, it's not designed as a fully fledged desktop distro.

The biggest driver of Linux adoption will most likely be further adoption and consolidation of web applications like *Microsoft Office* and *Adobe Photoshop*. Since these can be accessed via browsers, users are then free to switch to alternative platforms to access them.

It's also likely that continued adoption of *Docker* containers will allow more widespread use of DaaS (desktop as a service), meaning Linux will have greater support for proprietary programs. This could make a more convincing case for Windows and Mac OS users to switch.

Still, Linux has quite the mountain to climb, particularly given the persistent belief that it's difficult to set up and use.



The Linux desktop market share is currently 4.19%, more than twice what it was two years ago. It took 30 years to break 3%.

## DATE LINE

This raises the perennial question of whether 2024 (or some other arbitrary date) will be the 'Year of the Linux Desktop'.

CREDIT: StatCounter

DESKTOP

# KDE Plasma 6 released

The long anticipated upgrade to the Plasma desktop has been released with huge improvements.

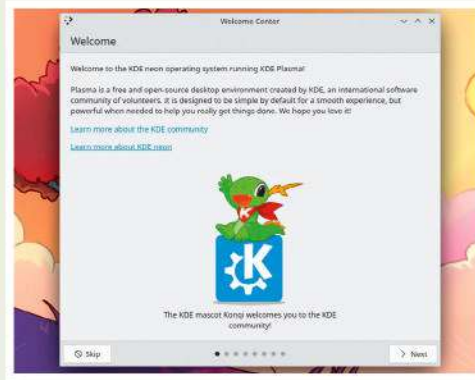
**O**n 28th February, Plasma 6 (built on top of Qt6) was released. One of the biggest changes is that Plasma now defaults to the Wayland display server, with experimental support for HDR. X11 will be supported for the foreseeable future and users can still select it upon login.

The desktop has also overhauled how it deals with workspaces. Mercifully for inept touchpad users, scrolling via the desktop no longer automatically switches users from one workspace to another. You can still use touchpad gestures to summon the new overview screen to display and select individual workspaces, however. The much-loved desktop 'cube' effect has also returned.

While we're talking mouse movements, for the first time Plasma now defaults to double-clicking for opening files and folders. (You can still switch back to single-click if you wish.)

The desktop still uses a tweaked version of the Breeze theme but now also incorporates a floating panel by default. It defloats when you're touching a window, including when said window is maximised.

Plasma also now ships with a new default sound theme, Ocean, which adds new sounds to



The new Plasma 6 desktop (shown running in KDE Neon) includes a number of tweaks to the theme and user interface.

improve usability, such as for connecting and disconnecting devices.

On 5th March, KDE released Plasma's first point release (6.0.1), which fixes various bugs including keeping files and folders visible as they're moved between screens. The new Overview effect has also been tweaked to allow closing via the top-left hot corner, while a bug has been fixed in the Plasma panel that caused it to disappear when users tried to add widgets.

[https://community.kde.org/Plasma/Plasma\\_6](https://community.kde.org/Plasma/Plasma_6)

GRAPHICS

# Nvidia moves to block ZLUDA

Ban on using CUDA translation layers in installed files.

**A**s more and more applications rely on the combination of Nvidia and CUDA, it was inevitable that third-parties would try to build on their success.

There's already examples of this, such as ZLUDA, a drop-in replacement for CUDA. The implications of running unaltered CUDA applications on non-Nvidia graphics cards could be very serious for the tech giant.

Perhaps it's unsurprising, therefore, that users wishing to install CUDA 11.6 onwards have to agree to a new clause in the EULA which states: "You may not reverse engineer, decompile or disassemble any portion of the output generated using SDK elements for the purpose of translating such output artifacts to target a non-Nvidia platform."

As commenters point out, it's not clear if the EULA is enforceable. Indeed, Nvidia's online licensing terms have forbidden running CUDA-based software on other platforms using translation layers since 2021 without taking obvious legal action against projects like ZLUDA.

Developers have also pointed out that in order to agree to the EULA, users must first install CUDA, so technically programmers could simply not reference or use it when creating their own translation layers.

Recompiling existing CUDA programs is also legal. AMD and Intel offer developers a way to do this via ROCm and OpenAPI respectively.

For now, CUDA has the edge, as ZLUDA doesn't fully support features like OptiX, but Nvidia's market dominance could reduce in time.

OPINION

# VECTOR VEXATIONS



**Dave Stokes** is a technology evangelist at Percona.

**“**Why does MySQL not support vectors in its community edition?

Generative AI is the hot topic in tech. GenAI relies on vector data. Yet Oracle has no plans to support vectors in the community edition of MySQL. If you want to try out vector data with other open source components, you have the option of MySQL HeatWave on Oracle's cloud service, or opting for another database entirely.

The general lack of new capabilities in the community edition is likely to force users to seek alternatives rather than sign up to an expensive subscription. This will put MySQL further behind options like PostgreSQL that already support vector data.

This is counter-productive for the MySQL community and for Oracle. Not having vector data support in the community edition will jeopardise the position of MySQL in the open source space. When you can use PostgreSQL and pg\_vector as a fully open source option, why pick something else?

There's already huge interest around MySQL to PostgreSQL conversion courses, in both the tech and the personal skills needed to be successful. This approach around vector data will only encourage more potential migration. In the meantime, we have to wait and see what the community does for itself.



## OPINION

## WINE FOR WAYLAND



**Alexandros Frantzis** is a senior software engineer at Collabora.

“2023 was a great year for the Wayland driver for Wine. The goal was to move forward from the experimental phase and make the driver a proper upstream component. A year later, after several merge requests, many people are now already able to use the latest Wine release to enjoy some of their favourite Windows applications in a completely X11-free environment.

So far, we have in upstream basic window management, such as full-screen, maximisation and resize, software rendering, Vulkan support, including Direct3D through WineD3D/Vulkan or DXVK, and basic support for HiDPI scaling.

While the recent release of Wine 9.0 did bring an experimental Wayland driver, our work is far from done. Our upstreaming efforts will continue in 2024, with a focus on multiple key features including emulation of display mode changes through compositor scaling, OpenGL support, improved positioning of transient windows (pop-ups, menus and so on) and even more window management.

If you're curious to see what all this progress looks like, there are several videos out there made by people using the Wayland driver (which is very exciting to see), so I'll let those speak for themselves. ”

## PLATFORMS

Processors in devices such as the first-gen Apple Watch still use the ARMv7 architecture but 32-bit developers are hard to find.



## FreeBSD moves to kill 32-bit

FreeBSD is deprecating 32-bit platforms; 32-bit binary support in 64-bit kernels will remain, however.

**F**reeBSD kernel hacker John Baldwin has announced that version 15.0 of the OS is unlikely to support ARMv6, i386 and PowerPC platforms. FreeBSD 16.0 will almost certainly not support ARMv7.

He did reassure readers that it will be possible to execute 32-bit binaries on 64-bit FreeBSD kernels and there are no plans to remove this feature from the stable branch.

Baldwin has justified this decision by pointing out that FreeBSD has also adapted to support various architectures based on

changes in the marketplace. While this initially meant supporting i386 systems, this has long since been superseded by x86\_64.

He also correctly stated that the pool of 32-bit developers is shrinking each year, with only the ARM7 architecture being used in recent system designs. The FreeBSD team may, however, support 32-bit architectures on a case by case basis.

More information is available via the FreeBSD 14.0 release notes: [www.freebsd.org/releases/14.0R/relnotes/](http://www.freebsd.org/releases/14.0R/relnotes/).

CREDIT: TechRadar

## OPERATING SYSTEMS

## PostmarketOS moves to Systemd

Phone OS has tentative support for Systemd.

**P**ostmarketOS (abbreviated as pmOS) is an operating system primarily for smartphones, based on Alpine Linux. One of the project's ultimate aims is to support the mainline Linux kernel on all phones.

In early March, pmOS took a step further towards this goal by providing a version of the OS that builds on top of Systemd.

At the time of writing, several-proof-of-concept images are available and developers are free to build their own version. Although Alpine Linux doesn't ship Systemd, there are currently no plans to move pmOS away from it.

PostmarketOS builds are now available using Systemd, even though its parent OS Alpine Linux doesn't ship with it.



## LAPTOPS

## Framework 16 laptop ships

Repairable, upgradable model costs under \$500.

**S**ince late February 2024, the first Framework 16 laptops have made it into customers' hands. The manufacturer's claims are bold: from just \$499 for the bare-bones model, you can have your very own customisable, repairable and upgradable laptop.

Framework devices largely follow open standards, including open source firmware. The new expansion bay can attach PCIe components such as the AMD Radeon RX 7700S, but this costs an extra \$400.

If you want a functioning laptop out of the box, prices start at \$1,399. Visit <https://framework.com> to learn more.



The Framework 16 offers even greater customisability due to supporting PCIe components, as well as a reasonable bare-bones price.

CREDIT: PostmarketOS Wiki, Framework Computer Inc



# Distro watch

What's behind the free software sofa?

## UBUNTU 24.04 LTS

Ubuntu 24.04 LTS Noble Numbat is currently scheduled for release on 25th April. At the time of writing, we're working with a daily build and expect a full review next issue. Nevertheless, we discovered that Canonical has announced this LTS release will be supported for up to 12 years, assuming users opt for an Ubuntu Pro subscription. The Ubuntu installer now defaults to Minimal, but you can still choose Full as well as third-party codecs. Learn more at <https://ubuntu.com>.



Ubuntu 24.04 will be supported for up to 12 years.

## FREEBSD 13.3

Despite moving away from 32-bit architectures in future versions (see *news story, opposite page*), this fourth release of the FreeBSD stable branch is still very much i386 compatible and brimming with updates. ZFS has been upgraded to OpenZFS 2.1.14. LLVM and the Clang compiler have also been updated to version 17.0.6. FreeBSD now ships with *OpenSSH* v9.6p1 and *Sendmail* v8.18.1. For a complete list of all the new FreeBSD features, pay a visit: [www.FreeBSD.org/releases/13.3R/relenotes/](http://www.FreeBSD.org/releases/13.3R/relenotes/).



FreeBSD 13.3 is dedicated to release engineer Glen Barber.

## LINUX FROM SCRATCH 12.1

The Linux From Scratch Project is both a Linux installation and a book that explains how to build a custom Linux system from scratch. For version 12.1, a total of 43 packages have been upgraded, such as *Binutils* 2.42 and *glibc* 2.39. A number of revisions have also been made to the book itself to improve readability. The Linux kernel has been updated to version 6.7.4, too. You can discover more at: <https://linuxfromscratch.org>.



The LFS book details how to build a custom Linux installation.

## KALI LINUX 2024.01

Beloved by penetration-testers everywhere, Kali's latest iteration, v2024.01, comes with a number of visual tweaks including the obligatory manual wallpaper and GRUB boot loader refresh. Kali is now easier to download as well, thanks to the addition of 32 new mirrors. As part of the continuing transition to GTK4 apps, the *Eye of Gnome* viewer has now been replaced by *Loupe*. The latest version of *Nautilus* also considerably speeds up file search times. Find out more at [www.kali.org](http://www.kali.org).



Kali comes with new tools and a refreshed theme.

## KDE NEON 20240228

This updated build of KDE Neon is still based on Ubuntu 22.04 LTS but includes brand new KDE releases including Plasma 6, KDE Frameworks 6 and all programs included with KDE Gear 24.02. Existing users can upgrade via Discover. New users can download the latest ISO or *Docker* images. There are some bugs in the latest so-called megarelease but developers assure users they're working on fixing them. Read more at <https://neon.kde.org>.



This User Edition of Neon includes all the latest KDE releases.

## OPINION

# FIXING OFFICE 365



**Italo Vignoli** is a founder of LibreOffice and the Document Foundation.

The European Data Protection Supervisor (EDPS), the independent authority for the protection of personal data and privacy within EU institutions and offices, has issued a press release stating that the use of *Office 365* violates rules in the GDPR, in particular the one regarding the transfer of personal data outside the EU.

The EDPS has decided that, by 9th December 2024, the transfer of data resulting from the use of *Office 365* to Microsoft and its affiliates must cease and the data processing must be brought back within the parameters defined by the GDPR.

As expected, the solution proposed is not to abandon *Office 365* and switch to *LibreOffice* – which would solve all data protection and privacy issues – but to intervene in *Office 365*, with the result that the solution will likely be inadequate and require further intervention.

Of course, the EDPS's decision should also affect institutions in EU countries, including schools, where data protection and privacy issues are even more serious as they affect children. We hope the EDPS will extend his decision to EU countries as soon as possible to avoid even greater damage than that caused at EU level.

And we hope this will lead, at least in some cases, to a migration from *Office 365* to *LibreOffice*.

## OPINION

## HAPPY BIRTHDAY!



**Jon Masters** is a kernel hacker who's been involved with Linux for over 22 years.

Canonical (the company that sponsors Ubuntu) just turned 20. Can you believe it? The Linux distro began life as 'no name yet' and there were rumours of secret meetings in Oxford to kick it off back in early 2004. I attended the first ever Ubuntu conference, which was really a large meeting held in the autumn of 2004 in Mataro, Spain. I recall having a lovely conversation or three with Mark Shuttleworth (someone with whom I would later compete when I joined Red Hat in 2006), and being impressed by the tremendous passion held by those within the community.

It was refreshing to see a Debian-derived 'commercial' friendly distro that ended up having legs. Some of us still fondly remember Progeny Linux, which was all too short-lived as an earlier attempt at building a commercial story with Debian. And in spite of my affiliation for all things Fedora, I always had a healthy respect for the work that went on within the Ubuntu community, and in particular for good friends like Jono Bacon, who did a lot of heavy lifting in the early years to build community (he would later go on to write a successful book on the subject). So, happy birthday Canonical, you're almost old enough to drink a toast where I live.

# Kernel Watch

**Jon Masters** keeps up with all the latest happenings in the Linux kernel, so you don't have to.

**L**inus Torvalds announced Linux 6.8, noting that the development cycle had been calm over the trailing couple of weeks, "just as it should be". The new kernel includes many performance enhancements under the bonnet. Among these are support for variable (multi) sized Transparent Huge Pages (THP), meaning that the kernel can now automatically handle memory using much larger pages while allowing multiple different huge page sizes, as opposed to the old-school approach of boot time selection (this is something that certain large workloads, such as databases, will enjoy). The new kernel also removes support for the venerable SLAB memory allocator, long since replaced by SLUB. An allocator manages 'arenas' of memory for easier data structure allocation.

More user-visible changes include support for Intel's Trusted Domain eXtensions (TDX) Confidential Compute (CoCo), at least on the host side. This means that (on suitable Intel hardware) 6.8 users can spin up encrypted guest VMs that are resistant to attack from a malicious hypervisor. Other architectures are in the process of gaining CoCo support. Another interesting change is the merging of an experimental driver with initial support for Intel's Xe graphics. This driver can be used on Tiger Lake and newer platforms, and is

slated to become the successor to the i915 driver. For now, you must explicitly enable it.

## Linux becomes its own CNA

One of the most interesting developments came in an email from Greg Kroah-Hartman (Greg K-H), in which he casually observed via a documentation patch, "The Linux kernel project now has the ability to assign CVEs to fixed issues, so document the process and how individual developers can get a CVE if one is not automatically assigned for their fixes." CVEs, or Common Vulnerabilities and Exposures, along with an associated CVSS scoring process is the industry standard for managing security vulnerabilities and tracking fixes to significant problems. CVE numbers can be issued by CVE Numbering Authorities (CNAs), of which Linux is now one, meaning that other entities can no longer issue CVEs against Linux.

The upstream patch adding the Linux CNA is opinionated, noting "Over time, CVE numbers were very often assigned in inappropriate ways and for inappropriate reasons." The patch talks about "ongoing abuses" and states that "the CVE assignment team are overly cautious and assign CVE numbers to any bug fix that they identify". Reactions were mixed, with a number of distro developers concerned that this could "DoS the system" (with the sheer number of CVEs) or "flood" distros with "low quality CVEs which have no analysis or scoring". **LXF**

## » ONGOING DEVELOPMENT

Pasha Tatashin posted "Dynamic Kernel Stacks", in a follow-up to a proposal from the LSF/MM Linux Kernel Memory Management conference. Traditionally, user applications use a magic stack that automatically grows (usually downward) as functions are called that perform local memory allocations. This is why you don't need to do explicit memory management for data on the local stack (the kernel is doing it on your behalf). But within the kernel, this is not the case, and so each task has a (small – for example, 16K) fixed-size stack. In the new world order, the kernel stack is still small, but not all of

that memory is allocated for every task until it is used, which is useful for very large systems.

Matthew Wilcox posted "Rosebush, a new hash table". Hash tables are very common data structures, known for their constant time access properties. As a result, hash tables are very widely used, yet not all hash tables are alike. The hash function used to generate indexes and other attributes, such as cache awareness, are critical to good performance at scale. Matthew claims Rosebush is cleverly optimised for cache friendliness. He does like his fun data structure names.



# Answers



**Neil Bothwick**  
learned to count in binary before decimal.

Got a burning question about open source or the kernel?  
Whatever your level, email it to [answers@linuxformat.com](mailto:answers@linuxformat.com)

## Wrong live user

Until Linux Mint 21.3, the Linux Mint live session user is **mint** and **\$HOME** is **/home/mint**. But in Linux Mint 21.3, the live session user is **linux** and **\$HOME** is **/home/linux**, though it is mentioned as **mint** on the official website. I am confused. How did I go wrong?

I have tried beta and stable versions of Linux Mint 21.3 Cinnamon, Mate and Xfce downloaded from <http://linuxmint.com>. In all three beta versions and three stable versions, the live session user is **linux**, the shell prompt is **linux@linux** and the home directory is **/home/linux** and **\$HOME** is **/home/linux**. There is no mint live session user or mint group. I have been using Linux Mint Cinnamon live sessions since 2009. Being a frequent traveller, practically, I live in live sessions. I have so many scripts as a mint user. The most intriguing factor is the anomaly. I have made some posts on this subject; but I could not get any answers. Kindly help me solve the mystery.

David Jebastian



This is indeed a mystery, because we just booted a freshly downloaded ISO of 21.3 Cinnamon and both the username and hostname were **mint**. Are you booting from the ISO copied to a USB stick or optical disc, or are you using a more customised setup? The reason for asking is that the GRUB menu specification includes this:

```
menuentry "Start Linux Mint 21.3  
Cinnamon 64-bit" --class linuxmint {  
    set gfxpayload=keep  
    linux /casper/vmlinuz boot=casper  
    username=mint hostname=mint quiet  
    splash --  
    initrd /casper/initrd.lz  
}
```

You can see that the username and hostname can be specified at boot time and default to **mint**. If you are using a custom setup, make the appropriate changes in your boot menu. If you are booting directly from a copy of the ISO, we are at a bit of a loss to see why this setting has changed for your system, but check the menu entry to see what is actually used. When the boot menu is on screen, press Tab to see the options used, and change them if necessary. You can also see the options after booting with this terminal command:

```
$ cat /proc/cmdline
```

You can also mount the ISO image from any normal distro session like this:

```
$ sudo mount -o loop /path/to/linuxmint-  
21.3-cinnamon-64bit.iso /mountpoint
```

However, this all raises another important point. You have hard-coded the username and home directory into your scripts. A more portable solution is to use environment variables in your scripts. The variables **\$USER**, **\$HOME** and **\$HOSTNAME** contain the username, home directory of your user and the hostname of the computer. You may also

find **\$PWD** useful; it contains the current directory that starts out as the directory from which the script was run.



## Wherever I hang my @

I recently purchased a Keychron K10 keyboard from the manufacturer's site. The advertising said that the keyboard was for Windows and Mac, and they also had a dedicated Linux user group. This user group is on Facebook, and I don't use Facebook, nor do I intend to open an account. It is a beautiful keyboard to type with, as long as you don't want any 'funny' characters. The @ sign and the euro symbol were very well hidden. I do some Bash scripts and Python scripts, and it's a nightmare.

I was wondering if you are aware of any Linux application that I could use to remap some of the keys. I tried VIA's web portal, but it refuses to play ball on Firefox. Any thoughts?

Sean Millar



This keyboard uses the **hid\_apple** driver module, which has a couple of options that affect the layout. The first sets whether the function keys behave as normal or whether you need to press the Fn key to get that behaviour. You can try the two options with this command:

```
$ echo X | sudo tee /sys/module/hid_  
apple/parameters/fnmode
```

Here, **X** is **1** to require Fn for 'normal' behaviour or **2** for Fn to give the special keys. There is also **0** to completely disable the Fn key, but you wouldn't normally want that. Once you know what you want, make the change permanent by creating the file **/etc/modprobe.d/hid\_apple.conf** containing the following:

```
options hid_apple fnmode=X
```

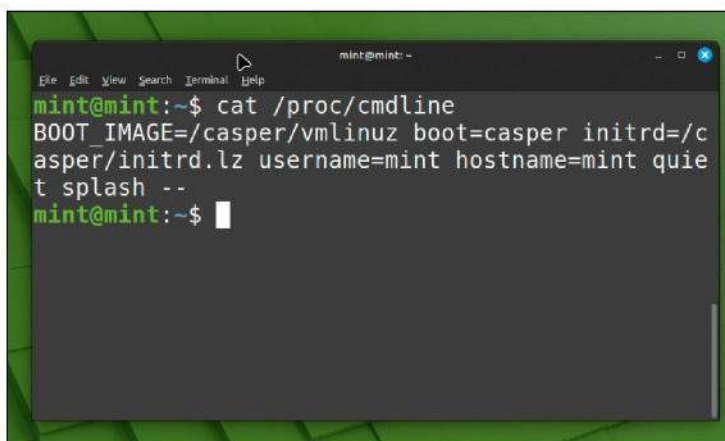
Then run:

```
$ sudo update-initramfs -u
```

The other option is **iso\_layout**, which can be **0** or **1**. Once again, you can test it temporarily with:

```
$ echo X | sudo tee /sys/module/hid_  
apple/parameters/iso_layout
```

Make it permanent by adding the following to **/etc/modprobe.d/hid\_apple.conf**.



Linux Mint live distros still boot with the user mint, but you can change this in the boot options.

conf and running `update-initramfs`:

```
options hid_apple iso_layout=X
```

There's a couple more options that you may like. To see them all, run:

```
$ modinfo hid_apple
```

To remap individual keys to suit your usage, you can use *Xmodmap*. First of all, create a configuration for the current layout with:

```
$ xmodmap -pke > ~/.Xmodmap
```

Then run:

```
$ xev -event keyboard
```

This shows the details of how each key is seen by X as you press and release it; you are interested in the keycodes. You can then edit the `~/.Xmodmap` file to remap keys. It contains lines like:

```
keycode 13 = 4 dollar 4 dollar EuroSign  
onequarter EuroSign
```

This shows what is returned by the key, first alone and then with the various modifiers. Edit this file, swapping around definitions to suit your needs and then apply the changes with:

```
$ xmodmap ~/.Xmodmap
```

Once you are happy with the setup you can make this change permanent. Try rebooting and see if your desktop picks up the `.Xmodmap` file, some do. If it doesn't, add this line to `~/.xinitrc` (create the file if it does not exist):

```
[[ -f ~/.Xmodmap ]] && xmodmap ~/.Xmodmap
```

This checks for the existence of the `.Xmodmap` file and loads it if found. There is a graphical editor for *Xmodmap*, called *XKeyCaps* ([www.jwz.org/xkeycaps](http://www.jwz.org/xkeycaps)), but it is very old and editing a text file is easy enough.

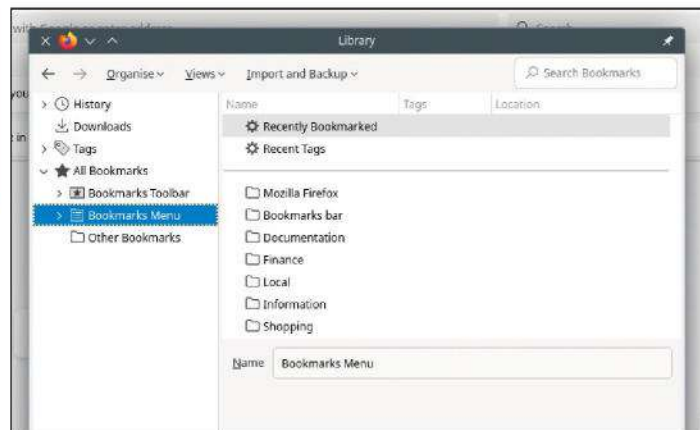
## Q Ye olde Firefox

How do I update *Mozilla Firefox* without losing my bookmarks?

I am running Ubuntu 17.04. Would the latest version of *Mozilla Firefox* run on my computer? It is 64-bit.

Ian Learmonth

Updating through your distro's package manager should not affect things like bookmarks, but you can back them up anyway.



**A** A normal update should not affect your bookmarks. As with most user-specific settings, bookmarks are stored within your **home** directory and software updates do not touch your **home** directory. That is, unless you are downloading *Firefox* from its website and installing it in your **home** directory, which is not a practice supported by your distro. In that case, you may want to back up your bookmarks before installing a new version. In *Firefox*, select **Bookmarks > Manage Bookmarks** from the menu (or press **Ctrl+Shift+O**). Here you can back up your bookmarks and restore them after an upgrade. You can also export bookmarks to an HTML file. That file can be imported into *Firefox* but also most other browsers; it is an easy way to transfer bookmarks between browsers.

However, you are running a seven-year-old version of Ubuntu, one that reached end-of-life some six years ago. That means you have not received security or software updates for almost that long. We would not be comfortable using such an old system online. Ubuntu does provide an upgrade path to update to a newer version, but it will have problems with a seven-year jump – one or two versions is considered safe. Your best option is a

reinstallation with the latest version. At the time of writing, that is version 23.10, but 24.04 will be imminent by the time this is published and that will be a long-term support (LTS) release. Standard releases are supported for only nine months whereas LTS versions have two years of updates and support, so it may be worth waiting for 24.04.

Before reinstalling, you should back up your **home** directory to an external device. Then you can copy it back once the new version is running. If the external device is formatted with a Linux filesystem, such as `ext4`, you can simply copy everything across because file metadata (permissions, ownership and so on) will be preserved. If the device is formatted with a Windows filesystem, this metadata would be lost with a straight copy, so you should use a backup program to copy and restore your files.

## Q Empty not empty

I am trying to delete a directory as root, but it won't go away.

```
$ rm -rf xyz/
```

```
rm: can't remove 'xyz': Directory not empty
```

```
$ ls -la xyz
```

```
$
```

## » A QUICK REFERENCE TO... FUSE

The Linux kernel supports many filesystems, both native to Linux and for other OSes. But there are many more specialised filesystems and it is unrealistic to expect the kernel developers to handle them all. Enter FUSE (Filesystem in Userspace), a module that allows external drivers to provide filesystem support. This opens a world of possibilities. The most

commonly used FUSE filesystem is probably `ntfs-3g`, which allows mounting and using of Windows NTFS devices. Yes, there is NTFS code in the kernel but the FUSE driver is preferred by many. FUSE can also be used to handle data transfer protocols, as with MTP. While this is not a filesystem specification, the FUSE drivers allow an MTP device

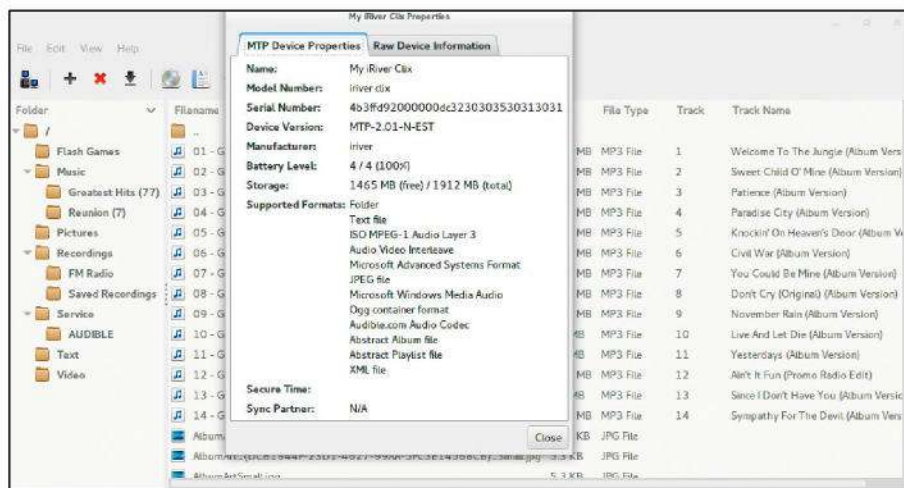
to be treated as a filesystem, extending the 'everything is a file' mantra of Linux.

Another popular FUSE driver is `sshfs`. This uses SSH to mount a remote filesystem, avoiding the need to set up Samba or NFS shares and open ports on your router, all with the security of SSH encryption.

Many cloud storage services supply software

for Windows and Mac OS to manage your files. A FUSE driver for a service allows Linux users the same degree of freedom, more so really, as you can use any file manager you prefer rather than being limited to their software. FUSE drivers have been written for most of the popular cloud services, and quite a few you have probably never heard of.





You can access the contents of an MTP device with GMP, or mount it as a filesystem with a FUSE program.

As you can see, *ls* reports the directory as empty. I am root, so it can't be permissions, so what is going on?

Bradley Howarth

**A** There's a number of possibilities here. Let's get the least likely, but worst, option out of the way first. You could be infected by a rootkit that is hiding files in there and has installed a modified *ls* to pretend they are not there. The easiest way to check this is to boot from a live distro and use *ls* from that to check. This is unlikely, but if it has happened, you should reinstall your distro.

Another possibility is filesystem corruption, which can be checked with *fsck*. The filesystem needs to be mounted read-only to run *fsck* fully, so if this is on your root filesystem, you need to boot a live distro to run *fsck* like this:

```
$ fsck -f /dev/sda1
```

Replace *sda1* with wherever your filesystem resides. A third possibility is that this is a mount point for another filesystem. Even if that filesystem is empty, it will not let you delete it, so *umount* first.

As you now realise, there are many possible causes, but a common and elusive one is that a deleted file is still locked. If you delete a file that is in use by an application, that file disappears from the directory listing but it is still there for as long as the application is using it. It is automatically deleted when the application releases it (or terminates). You can find such files with *lsf*:

```
$ lsf xyz | grep deleted
```

This shows any files that have been deleted but are still open. The first element on each line of output is the name of the process locking the file, the second is its process ID (PID). Knowing what is locking the file, you can go into the application and exit it to release the lock. If

that is not possible, maybe a background task has locked the file, you can pass the PID to kill to force it to close:

```
$ kill <PID>
```

If *lsf* shows the file is still locked, it is likely the process locking it has hung and is not responding to *kill*'s request to terminate, in which case you can force the issue with:

```
$ kill -KILL <PID>
```

Only use this as a last resort as it causes an unclean shutdown of the program. Alternatively, if you can't be bothered with all that, reboot the computer – a guaranteed, if somewhat brutal, way of releasing the locks.

## Music To Player

I bought a new MP3 player but it does not appear in disks and devices on my desktop. Neither do I see an icon to mount it. How am I supposed to get my music on to this hardware? My operating system is Debian 11.

Jamie Tyler

**A** Most media devices do not present themselves as USB mass storage devices these days; they use the Media Transfer Protocol (MTP), which is an extension of the Picture Transfer Protocol used by digital cameras. There is support for MTP devices in Linux, but we have found it to be a little flaky at times. If you check the settings menu on your devices, you may find an option to use mass storage instead of MTP. If you enable that, you should see an icon and have the device mounted when you plug it in. Otherwise you need to install an MTP filesystem driver, and there are several choices here, including *mtpfs* (<https://github.com/cjd/mtpfs>), *simple-mtpfs* (<https://github.com/phatina/simple-mtpfs>), *jmtfs* (<https://github.com/JasonFerrara/jmtfs>) and *go-mtpfs*

(<https://github.com/hanwen/go-mtpfs>).

We have used *mtpfs* and *jmtfs* with decent success. They are all FUSE filesystems and work in much the same way. For example, with *mtpfs*, run:

```
$ mtpfs ~/mtp
```

This mounts the first MTP device discovered at *~/mtp*, or wherever you choose to mount it. This assumes you have only one device attached, which is the usual scenario. If you are a glutton for punishment and want to work with multiple MTP devices, something like *simple-mtpfs* may be more suitable. With one device, the operation is the same as *mtpfs*. With multiple devices, where *X* is the number of the device from the list the first command gives you, you can do:

```
$ simple-mtpfs --list-devices
```

```
$ simple-mtpfs --device X ~/mtp
```

Whichever you use, once mounted you can access the contents from the terminal or your favourite file manager. As all of these filesystems use FUSE, they are all unmounted in the same way when you are finished transferring files:

```
$ fusermount -u ~/mtp
```

If you can't run these commands as your normal user, you could use *sudo* or check the ownership of the device created when you plug in the player; you may need to add your user to the relevant group.

There are graphical options – *GMP* (<https://gmp.sourceforge.io>) is one – as well as plugins for some file managers. Gnome users need to install *gvfs-mtp*, while in KDE you should see your devices in the Dolphin sidebar, as long as you have the *kio-extras* package installed. **LXF**

## GET HELP NOW!

We'd love to try to answer any questions you send to [answers@linuxformat.com](mailto:answers@linuxformat.com), no matter what the level. We've all been stuck before, so don't be shy. However, we're only human (although many suspect Neil is a robot), so it's important that you include as much information as you can. If something works on one distro but not another, tell us. If you get an error message, please tell us the exact message and precisely what you did to invoke it.

If you have, or suspect, a hardware problem, let us know about the hardware. Consider installing *hardinfo* or *lshw*. These programs list the hardware on your machine, so send us their output. If you're unwilling, or unable, to install these, run the following commands in a root terminal and send us the *system.txt* file, too:

```
uname -a > system.txt
```

```
lspci >> system.txt
```

```
lspci -vv >> system.txt
```

# Mailserver

## WRITE TO US

Do you have a burning Linux-related issue that you want to discuss? Write to us at *Linux Format*, Future Publishing, Quay House, The Ambury, Bath, BA1 1UA or email [letters@linuxformat.com](mailto:letters@linuxformat.com).

## LXF is wrong!

In the password manager article (*Roundup, LXF312*), it stated that you would still have to pay for premium features if you self-hosted *Bitwarden*. Just an update, but if you self-host with *Vaultwarden* (<https://github.com/dani-garcia/vaultwarden>), you get most premium features for free.

*Vaultwarden* still uses the official *Bitwarden* clients and is a much lighter-weight installation, better suited to home servers. Thanks for continuing to put out such a great magazine!

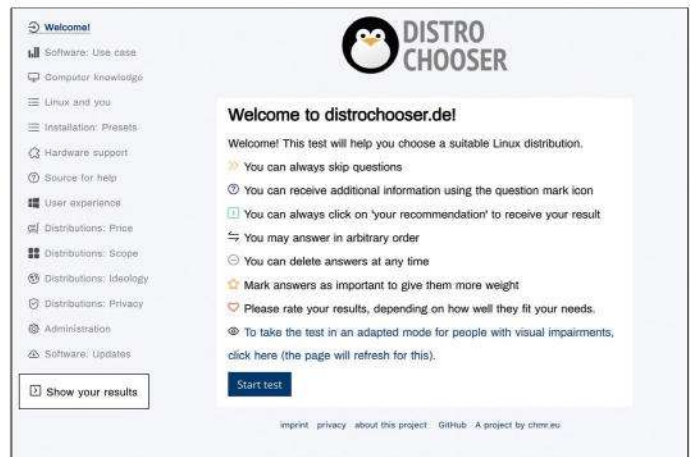
**Ken P**

Neil says...

Thanks for putting us straight on that one. I've not come across *Vaultwarden*, so that's good to know about for the self-hosting types. Personally, I'm a big fan of *Bitwarden* and have no issue paying the \$10 yearly fee to help fund it in some way. I can attest that moving from *LastPass* was utterly smooth, too, and if anything *Bitwarden* has a better web interface. And although the Android app could do with more polish, it's certainly as good as *LastPass*.

## We mean business

I'm looking for a Linux distro and a laptop model that is stable and will not start having problems after a couple



Come find the distro that some random internet person thinks is best for you!

of years. I'm looking for something that is business class and the budget is around £1,000. The idea is that the laptop should be suitable for multitasking and maintaining multiple terminals.

Currently, I'm working from Ubuntu running on a HP ProBook 450 G10 with an Intel Core i5-1335U and 32GB RAM. We are thinking of switching to Lenovo ThinkPads or the Dell Latitude series, but are open to any model and distro that will work well together.

**Hava Halime**

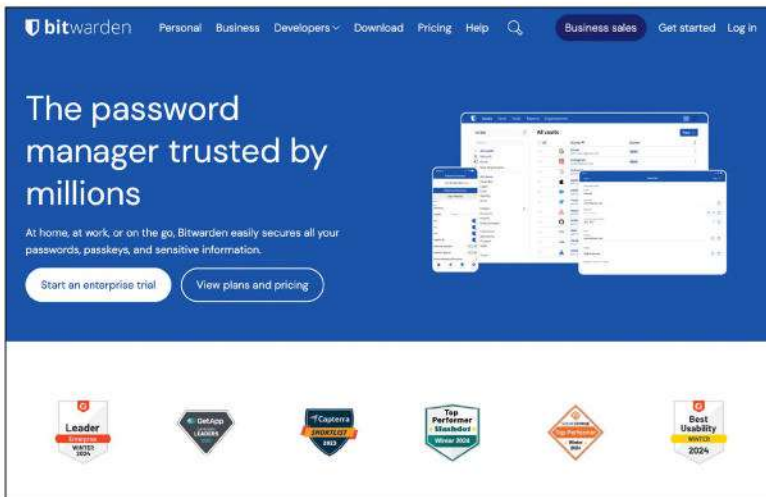
Neil says...

To be honest, that current laptop is pretty, erm, current. The processor was only launched early 2023, so it should barely even be a year old at this point. Not having tested any of those laptops, it's hard to officially recommend anything. We ran an article in *LXF313* on Framework gearing up to provide business-class laptops – as they're repairable, that's an interesting option, and they support Ubuntu and Fedora. No one got fired for buying Dell, is a saying, I believe, so have a look at the Dell models that officially support Ubuntu; the XPS 13 is the one to go for. Else Canonical runs a certified programme for Ubuntu that covers Dell, HP and Lenovo. See <https://ubuntu.com/certified/laptops>.

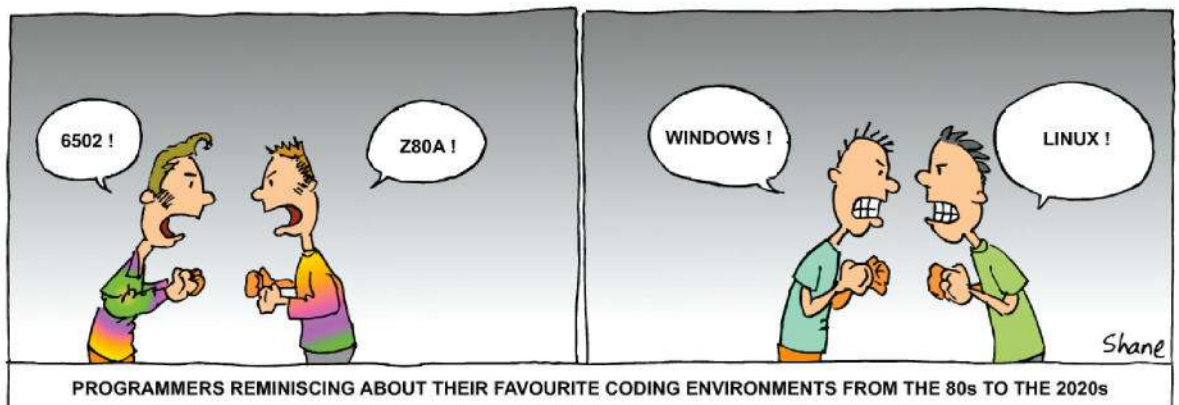
## Basic sales

I found it interesting that *Peachtree* accounting started out in Basic. I can also comment that I once worked for

Bitwarden is our go-to choice for password management and protection.



# Helpdex





the company that is now GameStop. As of 1998 when I left the company, its point-of-sales system was written in a dialect of Basic. I think QuickBasic, but when you rebooted the system (running MS-DOS, though the Basic variant wasn't GWBasic), you could observe some messages indicating it was launching some version of Basic to run.

**Bill**

Neil says...

It's always fun to hear people's experience. If you spot Linux out in the wild, let us know!

## One distro fits all

Honestly, I don't get this obsession with switching distros every other day! What's the obsession with having to reinstall everything and configure things from scratch – haven't people got anything better to do with their time?

**Sam Roberts**

Neil says...

Perhaps people like variety? No one's asking you to swap distros, it's just that some people like to experiment, and I'm assuming that many use a virtual machine to do so, so it's not actually that disruptive. I came across <https://distrochooser.de/en> recently, so you never know, it might help you find a better distro that you could try. **LXF**



■ We've tested the XPS over the years and it remains a solid bit of kit.

## » LETTER OF THE MONTH

### COSMAC forces

The article by David Bolton on Easy 6502 in the **LXF312** issue caught my attention. Back in the late 1970s, I was tasked with both learning assembler and writing a data acquisition program for our newly developed eddy correlation environmental flux machine called the Hydra. I had only programmed Fortran up to that point.

I was provided with an RCA COSMAC Microboard Computer with 8-inch floppy disks using the new RCA 1802 CMOS microprocessor and told I had not much more than six months to do it. Assembler programming teaches you not to be sloppy in your programming. The whole program took up 16 fan-folded pages. I even invented a magnetic tab A4-sized tablet that allowed me to stack what was being loaded-unloaded into/from the various data and program registers.

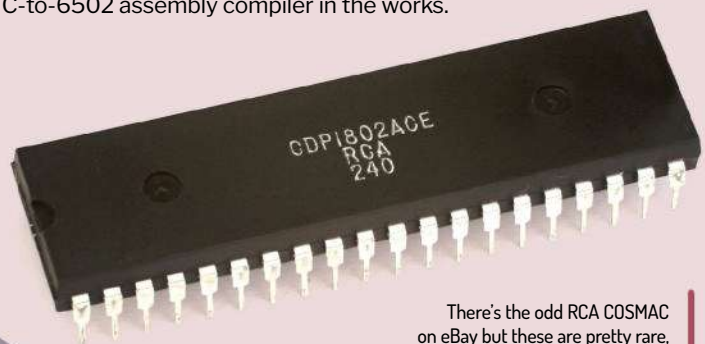
The system took sensor readings at 10Hz – we wanted to do it at 20Hz, but the speed of the processor and what it was being asked to do meant that even at 10Hz, getting all the data in and processing it took up 95ms of each 100ms cycle. Because of that I had to reserve a 100ms cycle at the end of each hourly data collection and processing period to work out all the various sums, means and variances. This was also our first experience with solid-state memory packs – a 16Kbyte CMOS paperback-sized module from GK Instruments, Milton Keynes.

You can find the paper here: [www.sciencedirect.com/science/article/abs/pii/0168192384900947](http://www.sciencedirect.com/science/article/abs/pii/0168192384900947)

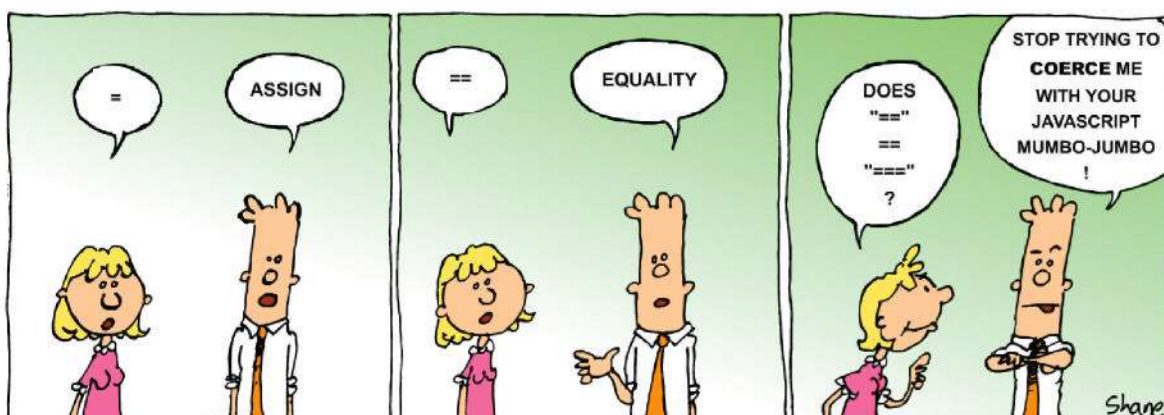
**Colin**

Neil says...

I love it when our readers let us know what they've been up to in the past (and present), so many thanks for taking the time to write in. David's going to write a bit more on 6502 down the line; we've got a C-to-6502 assembly compiler in the works.



There's the odd RCA COSMAC on eBay but these are pretty rare, powered by the RCA 1802 from 1972.



shane\_collinge@yahoo.com

**SUBSCRIBE** Save money today!

# SUBSCRIBE

and receive two

## 64GB USB sticks\*

included with every print subscription



### Five great reasons to subscribe:

- » Carry your data with you on two handy 64GB flash drives.
- » Never miss an issue of *Linux Format*.
- » Receive every issue delivered direct to your door before it's in stores.
- » Save 20% off the regular magazine price.
- » **NEW!** Full digital access to all prior and future issues when you subscribe to the print edition.

## SUBSCRIBE ONLINE AT

[www.magazinesdirect.com/LIN/C79L](http://www.magazinesdirect.com/LIN/C79L)

Or call **0330 333 1113** and quote **C79L**



Save money today! **SUBSCRIBE**



» **PLUS:** Exclusive access to the *Linux Format* subs area!\*

1,000s of DRM-free PDF back issues and articles! Get **instant access** back to issue 66 (May 2005) with tutorials, interviews, features and reviews. At **linuxformat.com**

## » CHOOSE YOUR PACKAGE!

Keep your subscription rolling!

**PLUS!**

New digital access with print subscriptions

### PRINT EDITION



**Only £16.75**

every 3 months  
3 issues of *Linux Format* in print  
every 3 months, saving 20%!  
With gift!

### DIGITAL EDITION



**Only £12.99**

every 6 months  
3 issues of *Linux Format* in digital  
every 3 months, saving 30%!  
Without gift!

**OUTSIDE THE UK?**  
Turn to page 59 for more great subscriber deals!

Terms and conditions: **Offer closes 30th April 2024.** Offer open to new print UK subscribers only. Please allow up to six weeks for delivery of your first subscription issue (up to eight weeks overseas). \*Your gift will be delivered separately within 60 days after your first payment has cleared. Gifts only available to subscribers on the UK mainland. Gift not available with a digital subscription. Gifts subject to availability. \*\*Access to *Linux Format* subs area only available to [www.magazinesdirect.com](http://www.magazinesdirect.com) subscribers. The full subscription rate is for 12 months (13 issues) and includes postage and packaging. If the magazine ordered changes frequency per annum, we will honour the number of issues paid for, not the term of the subscription. For full terms and conditions, visit [www.magazinesdirect.com/terms](http://www.magazinesdirect.com/terms). For enquiries please call: +44 (0) 330 333 1113. Lines are open Monday-Friday 9am-5pm UK time or email [help@magazinesdirect.com](mailto:help@magazinesdirect.com). Calls to 0330 numbers will be charged at no more than a national landline call, and may be included in your phone provider's call bundle.

# EXPLORE THE BEST LINUX & OPEN SOURCE SOFTWARE

From the best distros to system security, utilities and fun applications,  
we've rounded up everything you could ever need



Ordering is easy. Go online at:

**magazinesdirect.com**

Or get it from selected supermarkets & newsagents



# AMD Ryzen 5 8600G

The CPU of choice for **Chris Szewczyk** and his tight budgets.

## SPECS

**Socket:** AM5  
**Arch:** Zen 4 (Phoenix)  
**Process:** TSMC 4nm FinFET  
**Cores:** 6  
**Threads:** 12  
**Cache:** 384KB L1, 6MB L2, 16MB L3  
**Speed:** 4.3GHz (5.0GHz boost)  
**Unlocked:** Yes  
**GPU:** Radeon 760M  
**GPU clock:** 2.8GHz  
**Compute:** 8 units  
**AI:** Ryzen AI, 16 TOPS  
**Display:** Up to 4, DP 2.1, HDMI 2.1, AMD FreeSync, 8k @ 60Hz  
**Mem max:** 256GB  
**Mem speed:** DDR5 3,600-5,200, two-channel, no ECC  
**PCIe:** v4 20 lanes  
**TDP** 65W (45W-88W options)

If you're considering putting together a new system on a tight budget, but don't want to spend any of the money you do have on a current-generation graphics card, AMD's Ryzen 5 8600G may well be the solution for you. The 8600G is an affordable six-core variant of AMD's exciting Phoenix range of laptop, handheld and now desktop APUs. It doesn't have the grunt of its more expensive sibling, the £320 Ryzen 7 8700G, but it's still a capable 1080p gaming option that destroys everything else in its price range.

This is a six-core offering that's based on the Zen 4 architecture, but it has several key differences compared to its Ryzen 7000-series counterparts. For instance, the 8600G uses a monolithic die and it's built on a more advanced 4nm process. It's also got half the cache of the six-core Ryzen 7000-series CPUs, at 16MB versus 32MB.

It comes with a base clock of 4.3GHz, a boost clock of 5.0GHz and a 65W TDP. That's not dissimilar to the Ryzen 5 7600 – and, in fact, its base clock is a surprising 500MHz higher, though the 7600 has a 100MHz higher boost clock. Like the 8700G, the 8600G lacks PCIe 5.0 support and is configured with a PCIe 4.0 x8 connection for a graphics card.

## AI, aye?

A welcome, if for now untapped, addition to the 8600G is AMD's XDNA neural processing unit. As you've no doubt seen and read, AI is all the rage and local AI processing will potentially become more relevant as the software ecosystem matures. Who can say?

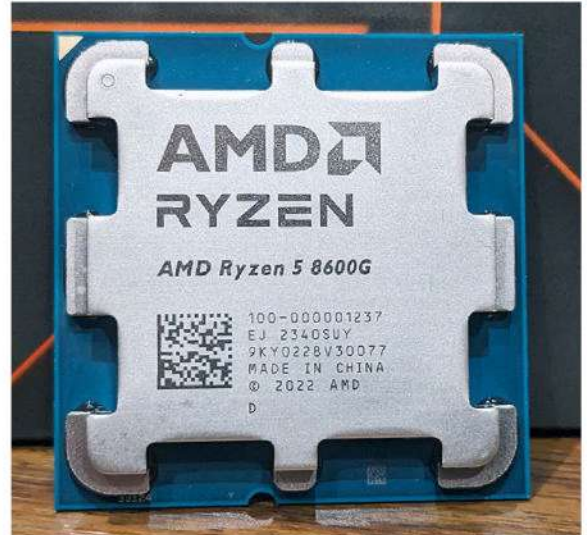
The integrated Radeon 760M is based on the RDNA 3 architecture. That puts it a full three generations ahead of the Vega graphics of the previous 5000G-series chips. On top of that, the 760M is clocked 800MHz higher than even the top-of-the-range eight-CU AMD Ryzen 7 5700G. On paper, the 8600G is faster. Much faster.

Given that AMD stopped bundling coolers with its 7000-series chips, the inclusion of AMD's Wraith Stealth CPU cooler is a nice little value add. It is on the small side, but the 8600G is not a particularly demanding chip.

Zen 4 chips are known to run hot. The 8600G never reached 80°C, which is a good result. We did use a 240mm cooler, which is a bit above the norm for this class of chip. The bundled Wraith Stealth would see temperatures higher than that.

Similarly, the power consumption results are a bright spot, coming in well under those of the 7600X. If you factor in the 8600G's efficiency, it's a solid option for a small form factor or media PC.

The 8600G's bright spot is its 1080p gaming performance. Even though it only has two thirds of the 8700G's shader count, its performance is good enough



Perfect for under-TV gaming and media PCs.

to make the claim that it can play games at 1080p with medium settings – if your threshold for 'playable' is 30fps on more demanding titles. Enabling FSR gives the 8600G a big performance boost.

The beauty of integrated graphics is that it's optional. There's nothing stopping you from installing your choice of GPU at any time. However, if you are planning to buy a discrete GPU at the same time as the rest of your system, a 7600 or any one of a number of Intel 12th, 13th or 14th-Gen options will deliver superior all-round performance. The 8600G cannot keep up with the more powerful 7600X. This is a result of the combination of the 8700G's lower TDP, lower long-run clock speeds and halved L3 cache.

The 8600G is a chip you should buy if you plan to make use of its integrated graphics. If not, there are better options available. However, if you care nothing for high core counts and transcoding videos every other day, the 8600G makes for an excellent general-purpose processor. **L3F**

## VERDICT

**DEVELOPER:** AMD  
**WEB:** [www.amd.com](http://www.amd.com)  
**PRICE:** £218

FEATURES	9/10	EASE OF USE	9/10
PERFORMANCE	8/10	VALUE	8/10

The best value desktop choice with integrated graphics, and with untapped AI co-processor power, too.

» **Rating 8/10**

# Deepin 23

**Nate Drake** is in too deep with Deepin – is this Chinese-made distribution a crouching tiger or a hidden dragon?

## IN BRIEF

Potential privacy issues aside, this is one of the most visually rich versions of Linux you'll ever use, with a host of excellent home-grown programs, such as the *Linglong* package manager.

## SPECS

**CPU:** 2GHz

**Mem:** 8GB

**HDD:** 64GB

**Buils:** x86-64,  
i386, RISCv64,  
ARM64,  
LoongArch64

**D**eepin development began way back in 2004 as Hiweed Linux. Until 2015, it was based on Ubuntu, but since then, successive versions of the OS have been based on the latest stable version of Debian (in this case, Debian 12 Bookworm).

The distro is known for its DDE (Deepin Desktop Environment), built on the Qt 6 toolkit (updated to version 6.6.1 for this release).

DDE is available in the repos of distros like Fedora as an alternative desktop, which is why you may have heard of this OS despite its main user base being in China.

Deepin is wholly owned by Deepin Technology, a subsidiary of Union Tech. The developers follow an irregular release cycle but aim for four releases per year.

Besides a graphically rich home-grown desktop environment, Deepin boasts its own window manager (*dde-kwin*), which for this latest development release contains a number of fixes for small bugs, such as one that previously caused window formatting not to follow those laid out in system settings.

Deepin 23 also includes the latest version of the OS's own package manager, *Linglong*, which includes an update to allow launching it immediately upon login.

The system repository itself has also received over 4,000 software package updates (including new packages). The OS now supports i386, riscv64 and LoongArch64 architectures.

Deepin's own *File Manager* has also been overhauled, with bug fixes that allow creating folders with spaces at the end of filenames, as well as the ability to copy folders directly from the Recycle Bin. Users can also create desktop shortcuts for *Google Chrome* with the relevant website icons.

Aside from a slew of its own packages and tools, according to other online reviews, Deepin comes preinstalled with proprietary software such as *Spotify* and *Steam*, which may explain the hefty 4.2GB ISO.

Deepin 23 also comes with UoS AI plugins, enabling you to use UoS AI via a free trial account.

If you do take the time to download the ISO, Deepin's own installer is extremely intuitive. The latest installer also now supports screen interface adaptation for 768p, 1080p, 2K and 4K resolutions. As snazzy and simple to use as it is, though, sadly there's no option to test Deepin in a live environment.

Post-install, however, we discovered why Deepin's interface has received such rave reviews online. The system first prompts you to create a user account and



DDE (Deepin Desktop Environment) may be resource-heavy, but it has style. It can also be installed in other distros, such as Fedora.

password before configuring your machine, then prompts you to log in to DDE.

The system also displays a helpful introductory video (in Chinese) showcasing all the latest features. English speakers can supplement their knowledge with the rather spartan *Deepin Manual* (available online or via the Deepin launcher). The project also has an English wiki (<https://wiki.deepin.org/en/home>).

Once the desktop loaded, we opened the launcher and were delighted to see that Deepin 23 has switched from the proprietary *WPS Office* to *LibreOffice* (7.4.2). We came across claims that Deepin comes with closed source software but this was absent when we checked.

When we fired up Deepin's App Store, however, we did find some proprietary options, including *Google Chrome* and *Steam*. In 2018, this same App Store drew controversy over the inclusion of statistics software that could potentially identify users, but this feature has been removed. Daniel Smith, head of threat research at Radware, has raised concerns over Deepin's updated statistical analysis service Umeng. **LXF**

## VERDICT

**DEVELOPER:** Deepin Technology

**WEB:** [www.deepin.org](http://www.deepin.org)

**LICENCE:** Mainly GPL

<b>FEATURES</b>	<b>9/10</b>	<b>EASE OF USE</b>	<b>8/10</b>
<b>PERFORMANCE</b>	<b>7/10</b>	<b>DOCUMENTATION</b>	<b>6/10</b>

A visual masterpiece and incredibly easy to set up. Let down by heavy system requirements and lingering privacy issues.

» **Rating 8/10**



# KaOS 2024.01

**Nate Drake** seeks to find some order in KaOS – a distro incorporating the KDE Plasma 6 desktop environment.

## IN BRIEF

KaOS is light on system resources but offers the best of Plasma. Bundled apps are well selected (except for an email client). The OS offers simple setup when it works.

## SPECS

**CPU:** 2GHz  
**Mem:** 1GB (2GB suggested)  
**HDD:** 8GB (25GB suggested)  
**Builds:** x86\_64

**K**aOS is an independently developed distro, though its design is inspired by Arch Linux. Originally released as KdeOS in 2013, it has since changed its name and follows a rolling release model.

The OS is designed solely for the x86\_64-bit architecture and exclusively uses Qt (currently version 6.6.1) as well as the latest version of KDE.

KaOS 2024.01 is the first stable release to incorporate the Plasma 6 desktop environment. It now also comes with an overhauled version of the OS's own Midna theme.

In the runup to release, KaOS developers have been shipping all ported applications from their Frameworks 6 branch into snapshot ISOs. Most bundled apps are now available as a Frameworks 6/Plasma 6 port. For those apps that are not ready, Frameworks 5 is still packaged, so they can be used in a Plasma 6 environment.

The display manager can run in Wayland mode, so KaOS is closer to being ready to move away from X11.

If you fire up the 2.7GB ISO, you'll also note that setup is handled by the user-friendly *Calamares* installer, which now supports automatic partitioning of popular filesystems like XFS, ext4, Btrfs and ZFS.

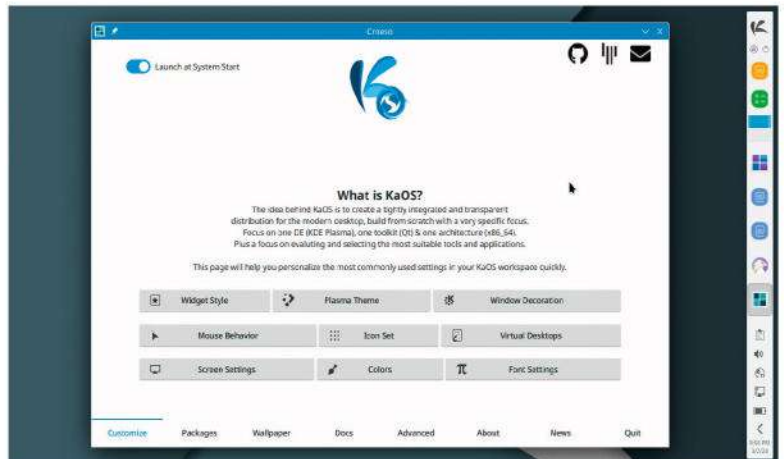
When we took KaOS for a test spin in a virtual machine, we were also pleased to see the OS can be booted in live mode.

On first launch, we also had a chance to view the built-in Guide, which is well laid out, although some of the grammar and spelling is dubious. The online documentation also contains a quaint warning against using a dial-up connection to download ISOs. During our tests in a virtual machine, the *Calamares* installer crashed when we tried to proceed with install.

Staying in live mode, we quickly moved to the *Application Launcher*. It's here we discovered that given that it's available as a pure Qt6/kf6 application, *LibreOffice* has now replaced *Calligra* as KaOS's default office suite.

If you do succeed with setup, KaOS's very own *Croeso* (Welsh for 'welcome') launches post-install. Written in QML, this handy tool lets you adjust some 15 commonly used settings, including viewing your distribution info, and select packages from no fewer than six different groups. From here, you can also customise your desktop wallpaper – KaOS comes with no fewer than 31 alternate backgrounds.

Browsing is handled by KDE's *Falkon* (v24.01.95). From the main desktop, you can also launch the Kate



The OS's own Croeso package manager enables you to adjust common settings such as the desktop background and configure package management.


advanced text editor, though on first boot we noticed a glitch where the icon was listed twice in the task bar.

Media lovers are also spoilt for choice in KaOS, which bundles not only the *Elisa* music player but *MPV*, *Harua* and (naturally) *VLC*.

Under the hood, the OS has been upgraded to FFMPEG 6. The kernel has also been updated to Linux 6.6.14. KaOS now incorporates *Systemd* 254.9, as well as *Python* 3.10.13, *Util-Linux* 2.39.3, *IWD* 2.13, *MariaDB* 11 and *Postgresql* 16.

KaOS also includes the OS's own *ISOWrite*, which, as the name suggests, can be used to write an ISO file to USB. Unlike previous versions, the tool also offers the ability to recover USB sticks used for this purpose.

If you want to install extra packages, you can use *Octopi*, a powerful *Pacman* front-end that KaOS has included instead of Plasma's *Discover*.

This is just as well, as despite the presence of chat clients like *NeoChat* and *Konversation*, there's no preinstalled email client. Luckily, you can use *Octopi* to install popular programs like *KMail* or *Thunderbird*. 

## VERDICT

**DEVELOPER:** KaOS  
**WEB:** <https://kaosx.us>  
**LICENCE:** Mainly GPL

<b>FEATURES</b>	<b>8/10</b>	<b>EASE OF USE</b>	<b>7/10</b>
<b>PERFORMANCE</b>	<b>8/10</b>	<b>DOCUMENTATION</b>	<b>5/10</b>

Excellent implementation of a pure Plasma 6 environment. Setup is simple but documentation needs an overhaul.

» **Rating 7/10**

# Bluestar Linux 6.7.6

**Nate Drake** finds blue is the colour, after firing up this intuitive and visually stunning Arch-based distro.

## IN BRIEF

Bluestar offers a beautiful UI, easy setup and a whole range of preinstalled apps. It's only let down by a lack of official website and documentation. Use at your own discretion.

## SPECS

**CPU:** 1GHz  
**Mem:** 1GB  
**HDD:** 20GB  
**Builds:** x86\_64

**B**luestar Linux's main goal seems to be to provide a user-friendly version of Arch Linux, on which it's based.

Its SourceForge page puts this in more colourful terms, stating it's "built with an understanding that people want ... a solid operating system that provides a breadth of functionality and ease of use without sacrificing aesthetics".

We mention SourceForge as, besides a rather active Facebook page, Bluestar stands alone in all the distros we've reviewed so far in not having a dedicated website. This is even more surprising given that successive stable versions have been released since 2016.

Still, if you do decide to download the weighty 5GB ISO from SourceForge, you can take the OS for a test spin in live mode. A lack of documentation makes it hard to gauge system requirements, though the ones listed (left) were adequate to run Bluestar in *VirtualBox*.

On first boot, users will notice that the desktop environment is KDE Plasma (v5.27.10).

We used the *Application Launcher* to launch the *Info Center*, where we discovered the OS also uses Qt (v5.15.12), KDE Frameworks (v5.115.0) and version 6.7.6-arch1 of the Linux kernel.

As stunning as the desktop is, Bluestar doesn't offer a particularly intuitive way to view installed apps. You have to move your mouse to a glowing blue bar at the top of the desktop to view the top panel. However, we did manage to find a shortcut to the *Application Launcher* via hitting the Super key.

The OS ships with the colourful *Latte* dock, though as main developer Jeff notes on the Bluestar Facebook page, development on this has been patchy. Luckily, he offers full instructions on how to switch to the more mainstream *Cairo* dock using the *Konsole* terminal.

If you do proceed with setup, this is handled by *Calamares*, which unlike plain Arch, offers a much more intuitive way to automatically partition the hard drive, encrypt the system partition and customise the look.

Once *Calamares* launches, you are offered a choice of Basic, Desktop, DeskPro and Developer installations. SourceForge mentions these are "each tailored to the needs of Linux users" but doesn't provide clarification. From reading other reviews, it seems these installs are centred around how advanced users are. For instance, the DeskPro version includes a live dashboard of system usage on the desktop care of *Konky*.

Speaking of the desktop, Bluestar offers no fewer than 40 alternative backgrounds. We immediately



For now, Bluestar comes with the *Latte* dock, although upgrade instructions for switching to *Cairo* are available via the project's Facebook page.

switched from the rather subdued default to the bright Altai wallpaper upon discovering this.

Having mastered the application launcher, we visited the Internet section to discover both the *Firefox* and *Konqueror* browsers come preinstalled. Bluestar also bundles the *Thunderbird* email client, as well as *Filezilla*, *Pidgin* and the *KTorrent* client.

The OS also comes with *LibreOffice* (v24.2.0), as well as the *Calibre* ebook manager. There's a number of multimedia apps, including the *Amarok* music player and *VLC*, though we felt that the inclusion of *Gnome CD Master* was a little redundant.

The OS's hefty install footprint is further explained by the presence of both *GIMP* and *Okular* for image/document viewing and editing.

The top panel also contains an icon of Blinky (the red ghost from *Pac-Man*), which you can use to launch the update tool. If you want to install more packages, in theory you can do this via *Octopi* (a graphical front-end for *Pacman*), but in our tests it crashed each time we tried to open it in the live environment. **LG**

## VERDICT

**DEVELOPER:** Jeff Hodd

**WEB:** <https://sourceforge.net/projects/bluestarlinux/> **LICENCE:** Mainly GPL

<b>FEATURES</b>	<b>9/10</b>
<b>PERFORMANCE</b>	<b>8/10</b>

<b>EASE OF USE</b>	<b>8/10</b>
<b>DOCUMENTATION</b>	<b>3/10</b>

Bluestar is, indeed, easy to use and aesthetically pleasing, but a Facebook page will never replace a dedicated wiki.

» **Rating 7/10**



# Drauger OS 7.6

**Nate Drake** puts this distribution through its paces – is Drauger a gaming dragon or just a plain drag?

## IN BRIEF

We love the idea of a gaming-specific version of Linux, but Drauger fails to deliver. Given the setup/kernel issues, you're far better off installing Ubuntu, then adding gaming-specific packages yourself.

## SPECS

**CPU:** 1.8GHz (2GHz suggested)  
**Mem:** 1GB (6GB suggested)  
**HDD:** 32GB (128GB suggested)  
**Builds:** x86\_64

In Norse mythology, draugrs are undead warriors with a penchant for the blood of the living. *Skrym* players will also know they're a common foe you encounter when dungeon-crawling, so developer Thomas Castleman, a keen player, named this OS accordingly.

One typo later, Drauger OS was made available in 2018. It's based on Ubuntu LTS (in this case Ubuntu 22.04 Jammy Jellyfish) and is supposedly optimised for gaming.

The main website's About page is also very clear on what Drauger is not: if you're thinking of installing Drauger as your daily driver, you'll be disappointed. There's no preinstalled email client, office suite, BitTorrent client or any apps you'd need when normally using Ubuntu.

Previous versions of Drauger used the xanmod kernel, which is ideal for gaming, but this iteration apparently uses a version that is compiled in-house. It would be more correct here to say "versions" as Drauger uses kernel 6.2.9 in the live environment but switches to version 6.6.11 once installed.

Post-install, this triggered a kernel panic in our test virtual machine, which we resolved by rebooting and choosing 6.2.9 from Drauger's Advanced Boot Options.

On first login, you'll notice Ubuntu's default Gnome desktop environment has been swapped out for Xfce, along with a dark Nocturn GTK3 theme. We were impressed by the Drauger welcome guide, which walks you through various sections of the desktop.

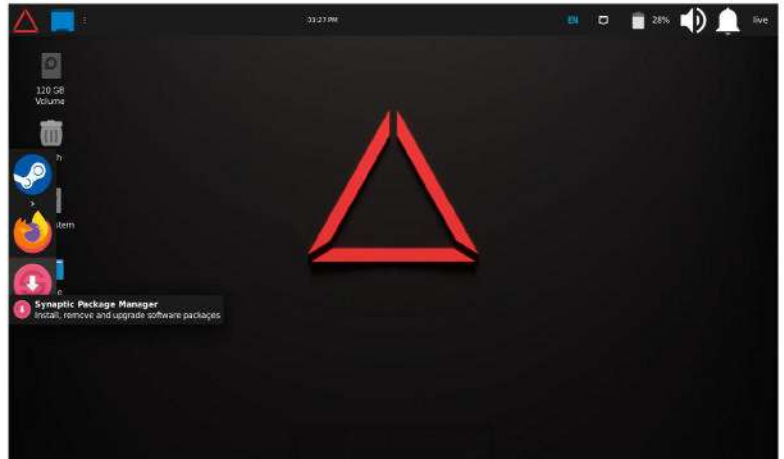
This includes a whistle-stop tour of the panels at the top, left and bottom of the screen. The left bar contains launchers for some of the preinstalled apps, including *Steam* and the *Synaptic* package manager. Web browsing is handled by the Flatpak version of *Firefox* (v110.0.1), which can be launched from here.

In the live environment, we felt this placement was a design flaw, as it eclipsed desktop icons like Home and Trash, but these had vanished post-install.

The bottom panel allows switching between virtual desktops, though we found it hard to distinguish from the default wallpaper. Luckily there are six alternatives.

If you are willing to risk kernel panic and proceed with setup, Drauger ships with its own *System-installer*. The interface for this is dated, where you have to click individual buttons to launch windows and configure settings like language and keyboard. A more intuitive installer like *Calamares* would be a much better choice.

Once the OS loaded, we decided to explore the gaming-specific features of Drauger. The main website



The left-hand panel contains launchers for a number of apps, including Steam, Firefox and Synaptic. In live mode, this obscures the desktop icons.

is keen to stress that the OS ships only with open source software, although the presence of the *Steam* client for Linux seems to contradict this.

Other gaming specific programs include *GameHub*, *Heroic Games Launcher* and *Lutris*. We fired up *Lutris* and used the search feature to download and install the 1993 shareware version of *Doom*, which ran, albeit somewhat jerkily.

Drauger doesn't seem to assign a dedicated swap space to the drive during automatic partitioning. After checking the root drive immediately after installation, we found that instead it creates a **.swapfile**, which was already almost 12GB. This may account for Drauger's large install footprint (17GB) – over twice that of our Ubuntu LTS VM.

Drauger does come with *Gnome Software* and even ostensible support for Flatpaks, but when we tried to install two apps via Flathub (*VLC* and *Polari*), we encountered an **unmet dependencies** error. Installation via the command line and *Synaptic* via Ubuntu repositories worked like a charm, however. **LXF**

## VERDICT

**DEVELOPER:** Drauger OS Development/  
 Thomas Castleman **WEB:** <https://draugeros.org>  
**LICENCE:** Mainly GPL

<b>FEATURES</b>	<b>5/10</b>	<b>EASE OF USE</b>	<b>6/10</b>
<b>PERFORMANCE</b>	<b>6/10</b>	<b>DOCUMENTATION</b>	<b>7/10</b>

With its custom kernel and installer, Drauger feels much like a work in progress rather than a fully fledged OS.

» **Rating 6/10**

# Roundup

TimeCamp » Timewarrior » Hamster  
» Arbtt » TimeSlotTracker



**Michael Reed**

says he's spent 74.5 hours writing about Linux so far this year. He's full of fascinating facts like that.

## Time trackers

This month, **Michael Reed** is keeping a strict eye on himself and trying to improve his work efficiency by investigating time-tracking applications.

### HOW WE TESTED...

We installed each of the time-tracking applications on a desktop PC running an up-to-date Linux Mint installation with Xfce as the desktop. Where double-checking things such as installation issues was needed, we also installed on an Ubuntu 23.10 machine.

We used the time trackers while carrying out our work to see how easy to use they were on a day-to-day basis and to see how useful the activity logging was. This included applying automatic tracking features where available.

We tried to get into the more advanced features where extensions and plugins were supported on a particular piece of software.

Once we had some recorded data that we could play with, we explored the report-generation possibilities. Pretty charts, where available, were appreciated.

Finally, once we had the Linux-based software up and running, we investigated the possibilities for moving between machines and environments while continuing to track our time.



**T**his month, we're examining five of the best time-tracking applications available on Linux. These are tools that are useful to freelancers, office workers (with a paranoid boss) or simply people who like stats and want to know how they actually spend their time.

*Timewarrior* may appeal to the text-mode traditionalist as it's a purely command-line-driven application. However, it can output some natty looking text-mode charts and it's a good choice for custom analysis of its data.

*Arbtt* is another 'Linuxy' piece of software as it's operated from the command line. The main part of the software automatically

monitors the user's activities for hyper-accurate recording of task activity. Like *Timewarrior*, it provides good opportunities for the stats obsessive who wants to use Python code to analyse activity data.

*Hamster* is a conventional desktop application with a user interface that fits in well with the stylings of the Gnome desktop.

*TimeSlotTracker* is a multiplatform Java desktop app, and it's quite feature rich.

The other applications are open source and completely free, but we also added *TimeCamp*, a freemium service, into the mix. We wanted to see how useful a free account was compared to some of the other options.



# Linux native application

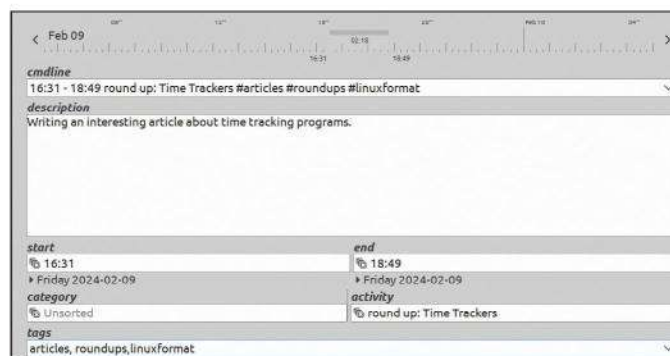
Other options are nice, but we expect a well-integrated Linux app.

**T**he *TimeCamp* app places itself in the status panel area of the desktop. It offers a hybrid between a native app and a web-based one. For example, several functions open a page in your browser. However, it does provide some advantages of a native app as the usage reports contain detailed records of your actual desktop use, showing which apps and browser windows you were using at different times, something that wouldn't be possible from a purely web-based tracker.

It is possible to start and stop app tracking while offline, but the online documentation makes it clear that it is your job to select the synchronisation function in the app once back online and that some of the analytics data is not acquired in this mode.

*Timewarrior* is operated from the command line and offers text-mode output. It consists of a daemon that runs in the background along with a collection of executables. Much use is made of command-line options to display captured data. We're not going to mark it down due to it being a text-mode tool, as many Linux users prefer this, and there even are a few advantages to a system that can be fully automated from scripts.

*Arbtt* is a command-line text-mode app. In fact, it's made up of a selection of command-line tools. The online documentation advises to set things up so the tracking daemon, *arbtt-capture*, is



As a Linux native GUI application, Hamster has some advantages when it comes to speed and ease of editing time entries. It takes a couple of clicks to amend an entry.

run when the desktop starts up. Configuration is carried out by editing a text file, saving it and restarting *arbtt-capture*.

*Hamster* is a fairly simple Python GUI application that fits in well on a Gnome desktop, but it's usable on any Linux desktop.

As it's a Java app, *TimeSlotTracker* has many of the characteristics of a native desktop app, such as pull-down and pop-up menus, and it features a status panel icon. Looks-wise, it doesn't fully fit in on a Linux desktop, but it is launchable from the standard application launcher menus if installed as a DEB file.

## VERDICT

TIMECAMP	6/10	ARBTT	7/10
TIMEWARRIOR	8/10	TIMESLOTTRACKER	7/10
HAMSTER	8/10		

The TimeCamp Linux app is fairly simple. TimeSlot is the most fully featured, but it's a Java application.

# Installation complications

The first hurdle is always obtaining and installing the software.

**T**he Linux Mint system package for *Timewarrior* was out of date, so we installed the latest version from Snap.

The *Hamster* package in the distribution repository was the latest stable release (which is quite old), but we could also have installed via Flatpak, Snap or built from source code. There are installation instructions for Debian-based distros such as Ubuntu along with Fedora and OpenSUSE.

The system package for *Arbtt* was out of date. It wasn't available on Flathub or Snap, so we had to build it from the source archive. The build process involved using *Cabal*, a package manager for the Haskell programming language. We'd rate it at the lower end of medium difficulty as build jobs go. Once the program is actually installed, a default configuration file has to be created by copying it from the documentation, and the program is invoked from the command line.

We had a few problems installing the native Linux *TimeCamp* app on both Linux Mint and Ubuntu after having downloaded the official packages from the website. At one point, we had to manually install a dependency it needed. These dependency



The Arbtt build instructions work perfectly well, but building from the Git repository means that it's up to you to keep checking for updates and rebuilding.

problems affected the DEB package and the ApptImage version. Overall, it had the feel of a package that hadn't been maintained properly, but we did, eventually, get it to install and work.

On its SourceForge page, *Timewarrior* is available as a multiplatform JAR file and as a DEB file. It was not present in the repositories of Ubuntu or Linux Mint.

## VERDICT

TIMECAMP	6/10	ARBTT	5/10
TIMEWARRIOR	7/10	TIMESLOTTRACKER	7/10
HAMSTER	8/10		

None of the systems excel in terms of variety of available packages, but Hamster is most widely available.

# User interface design and use

## The ease of daily use.

If the application is fiddly and difficult to use, you are less likely to be meticulous when it comes to logging actions. In the same way, having to put in extra work every time you recommence your activity can make getting back to work even more of a chore than it usually is.

We're particularly interested in the daily use cycle of these applications. That is, how easy is it to start or stop logging, the principle function of a time-tracking application? Beyond that, we expect other day-to-day features to be accessible and quick to display. We appreciate a real-time update of how long we've been working on the current job.

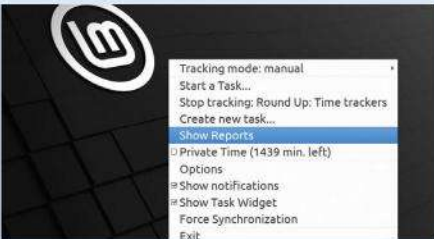
Editing an entry is another commonly used function and we want that to be made as simple as possible. Other functions, such as creating a report or summary, are used less often but, saying that, we don't want to have to jump through hoops to get it done.

### TimeCamp

6/10

Most of the *TimeCamp* user interface resides in the status panel app, meaning that there isn't anything in the way of a real-time task clock. Some of the options open a browser page, leading to a slightly disjointed experience. However, the workflow for creating a new task or resuming a task is problem-free and can be carried out from within the native Linux application. We place a higher priority on the smoothness of that area of the interface than any of the others. Actions such as amending an entry have to be done via the website.

Sticking with the native interface, some areas used a font that was too small, but it was usable. Overall, we were glad to see a Linux native application, and it does offer some important functionality in the way of tracking. However, for actual use, we consider the web interface to be superior, particularly as you're going to have to visit it from time to time anyway.



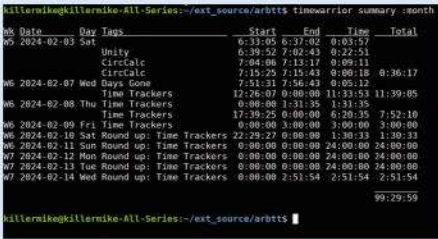
### Timewarrior

6/10

*Timewarrior* is a pure command-line app. Some Linux users will like the exactness of typing in commands to activate or stop the tracking. However, we found it a grind as we were used to the GUIs of the other options. As command-line apps go, we're fans of those that allow tab completion when building up the entire command, but *Timewarrior* doesn't offer that feature.

As it's command-based, there's a learning curve as you discover the options to do things such as start an activity or amend a previously logged activity. The command-based approach means you can carry out an amendment, such as adding or removing time, to multiple activities with a single command.

It's a complex system that takes a bit longer to learn than some of the other applications, but once you've picked up the basic command options, day-to-day use is acceptable and even offers a few advantages as it's scriptable.



## Extra features

Beyond the basic options, what else can the software do?

*TimeSlotTracker* can add custom attributes to a task or a time slot in the form of key/value pairs, so you can attach almost any type of info to a task or activity.

*Timewarrior* has integration features with *Taskwarrior*, a command-line text-mode task and to-do list manager. The configuration involves some work at the command line to copy files. Once up and running, when a task is started in *Taskwarrior*, *Timewarrior* begins tracking the time spent on that task.

It also has an extension system that uses Python plugins. Searching online, we found a number of intriguing extensions. So, although *Timewarrior* doesn't have billing facilities as standard, they can be added via an extension. Many of the extensions also improve the report generation facilities. The website lists 95 extensions, which are mostly well maintained. However, they involve manual installation and configuration.

As applications go, *Hamster* keeps things plain and simple. It does benefit from a KDE plasmoid, however, which provides an extensive front-end that fits into the KDE desktop.

*Arbtt's* documentation links to a couple of contributed tools that offer extra report formats. It can output as CSV for further processing and has support for R, a statistical analysis language.

We were sometimes a bit annoyed that the *TimeCamp* web interface presented us with a page telling us about the upgrade options to the paid version. In actual fact, the free-for-life plan has a lot of features that aren't immediately apparent, and you can ignore the landing page and simply select features from the sidebar. In particular, we almost didn't see some of the report and charting features that are available.

### VERDICT

TIMECAMP	7/10	ARBTT	4/10
TIMEWARRIOR	8/10	TIMESLOTTRACKER	4/10
HAMSTER	4/10		

*Timewarrior's* extensions offer a lot of opportunities to add extra features. It's worth exploring the *TimeCamp* web interface to see what it can do.

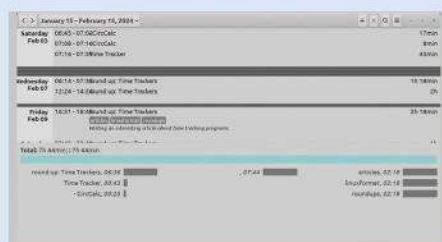


**Hamster****7/10****Arbtt****6/10****TimeSlotTracker****8/10**

In keeping with the modern Gnome style, *Hamster* locates some of its controls in the title bar area of the window. There is a large area showing the currently summarised period and icons for starting or stopping a task, and calling up the menu and icons for the main functions can be carried out quickly and efficiently.

When a new activity is started, a pop-up dialog appears, and the drop-down menu means it's reasonably hassle-free to continue a previously established task.

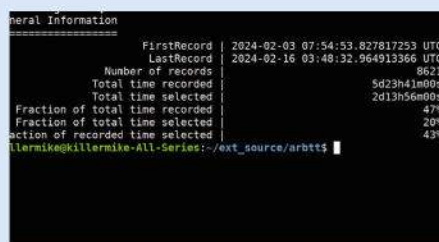
The main area of the window is taken up with a summary of the given period, providing a good sense of context. A double-click on previous tasks brings up the editor dialog for amendments, which are easy to make. A large bar on the bottom of the window splits the screen to show some statistics. The only criticism of the interface we'd make is that some of the elements are a bit non-standard and require experimentation to figure out.



*Arbtt*, like *Timewarrior*, is a command-line application. Because it is an automatic tracker, there is less interaction with this interface while working than with other programs. The application is actually a collection of smaller programs that begin with the *Arbtt*- prefix.

To get the best out of the program, you have to work on the configuration. As with most command-line programs, this is carried out by editing a text file. In this case, it has quite a complicated Haskell-inspired layout and syntax.

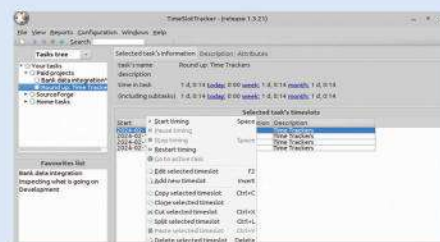
There's a paradox with *Arbtt* in that it eventually requires almost no interaction in day-to-day activity recording, but you have to work with the documentation and wrestle with the main configuration text file to get it to that state. Most of the other programs are more open to an approach where you can explore the user interface and try things out to see how the program works.



Despite being a Java Swing GUI app, *TimeSlotTracker* looks reasonably integrated on a Linux desktop when a theme such as GTK+ is selected. It's a drag-and-drop interface when it comes to reordering items, and most of the UI works like a typical desktop application. There is a status area icon, and it offers a few simple features.

Turning to the main interface, it's a multi-pane GUI. The left-hand sidebar contains a hierarchical list of tasks that you must add to before you can begin tracking. The icon bar along the top has video-style controls for controlling the task monitoring, along with a real-time search bar. There is another panel to show detailed information on the selected task.

This is a featureful interface that has the benefits of a desktop app in terms of responsiveness. It's the fastest and most convenient of all of the systems for tasks such as amending or copying time slots.



# Automatic tracking

Insights from hands-free analysis of your activity can often be fascinating.

**T**imeSlotTracker has activity monitoring and this includes an optional facility to take screenshots and place them in a user-defined directory. However, this isn't the type of intricate user monitoring that some of the other options offer. Idle detection (*what do you mean?—Ed*), another useful feature, is a separate option.

Thanks to *TimeCamp*'s native Linux app, it can track user activity such as application and website usage, such as how much time you've spent on a specific email in GMail or a specific document in Google Docs. Some of this activity data is accessible in the various report and summary modes. However, some of the more advanced summary data specific to this type of data is only available on the *TimeCamp* paid tier.

*Arbtt* is specifically designed to capture user activity data in an automatic fashion. There is quite a lot of refinement available though configuration via a text file. It's possible to set it up so that it looks for certain strings in the application title and assigns certain tags to certain activities. For instance, the

official documentation offers examples that can assign the 'social media' tag to the log when certain sites are being visited. However, like much of *Arbtt*, this type of detailed tracking requires quite a lot of work to set up in the configuration file.

*Hamster* lacks automatic time-tracking facilities.

*Timewarrior* has a kind of automatic tracking. It enables you to define periods when you know that you will be engaged with tasks and exclusion periods. It takes quite a lot of setting up, however, and it's limited in what it can do compared to *TimeCamp* and *Arbtt*.

## VERDICT

TIMECAMP	8/10	ARBTT	7/10
TIMEWARRIOR	5/10	TIMESLOTTRACKER	6/10
HAMSTER	N/A		

**Arbtt can do a lot in this area, but it takes work to set it up. TimeCamp is useful from right the start.**

# Cloud options and alternative platforms

Working away from the main computer.

As it's a purely command-line-based tool, *Timewarrior* can be used over SSH, which means that it can be accessed over a network. This could also be extended to a mobile device running an SSH client. However, the host machine would have to be set up and available over the network and left running, and you're restricted to a command-line interface when starting and ending task recording, which could be awkward, particularly on mobile. It is only available to run on Unix-type operating systems such as Linux.

If anything, *TimeCamp* is a stronger application when accessed from the website. Every function can be accessed on the web and many functions can only be accessed in that way. This means that it can be accessed from multiple machines and even from a mobile device. Speaking of which, there are also dedicated mobile phone apps for Android and iOS. This is along with applications for Linux, Windows and Mac OS.

As *Arbtt* gathers information about usage from the computer on which it is run, it can't be used remotely. In addition to various Linux versions, there is a Windows version.

*TimeSlotTracker* is a multi-platform application, and this means it can run on Windows and Mac OS machines as well as Linux. Unfortunately, there isn't a mobile app version, and there



*TimeCamp* can be operated from the mobile phone app, and it's a fairly extensive piece of software. As well as the starting and stopping of tasks, it also has some charting and summary modes.

are no cloud data-sharing features. What could work is exporting data from the program on one computer, then importing the data back into the program on another.

*Hamster* is a simple GUI-based time tracker that is designed to be used on a single machine.

VERDICT			
TIMECAMP	9/10	ARBTT	4/10
TIMEWARRIOR	6/10	TIMESLOTTRACKER	5/10
HAMSTER	N/A		

TimeCamp lives happily in the cloud or on various native platforms.

# Reports and summaries

For some, this is one of the most important parts of the system.

*TimeWarrior* has a couple of report modes. The **summary** command option outputs a summary of total time taken on different tasks. There are quite a few options that can be applied to this, such as limiting the summary to a period or filtering by tag. It also has a graphical text-mode charting option. There is also an extension system that uses Python plugins for the ultimate in custom data processing.

Even on the free tier, *TimeCamp* has a full set of report features. Many are based on business needs, and can help you with billing by assigning hourly rates for different jobs. There are some helpful charting facilities that help to give an overall view.

*Hamster's* report generator is accessed via Export on the main menu. The report encompasses the period specified in the main interface. It is a self-contained HTML file that looks good in itself. It can be customised and a custom template file can be created from the default one, but it's more work than being able to tweak the report parameters and layout from the GUI.

*Arbtt-stats* produces the summary for *Arbtt*. As *Arbtt* is an automatic system, how detailed this information is depends largely on the amount of work you put into configuring *Arbtt*. For



*Timewarrior* is limited to text mode, but it can produce coloured charts. You can specify a period as day, week or year, controlled via command-line options.

example, by default, the output shows how much time you have used a program such as *Chrome*, but you have to configure *Arbtt* to assign a custom tag to windows that have Google Docs to see how much time you spent with that individual website.

*TimeSlotTracker* has a well-organised selection of report types, and it has the advantage that it can output to a number of formats such as CSV in addition to fully formatted HTML.

VERDICT			
TIMECAMP	8/10	ARBTT	6/10
TIMEWARRIOR	7/10	TIMESLOTTRACKER	7/10
HAMSTER	6/10		

TimeCamp has a lot of reporting facilities even on the free tier. The others have at least some facilities, too.



# The verdict

## Time trackers

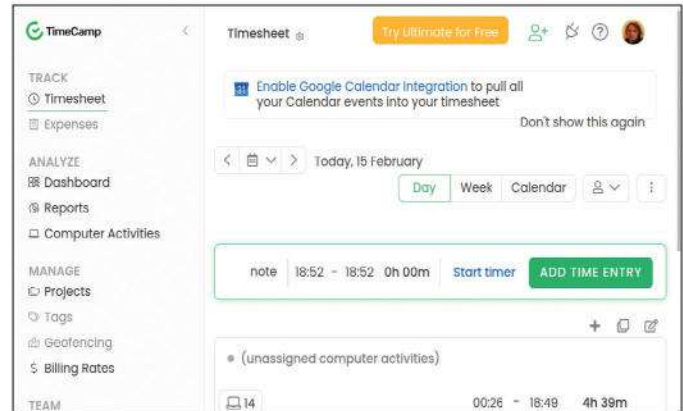
**W**e're choosing *TimeCamp*, a freemium solution that isn't open source, as the overall winner. The facilities on the free tier are decent, and the free package could be everything that a single user would ever need. It can be used from any platform as it is accessible from the web. In addition, there is a mobile phone app. The native Linux application isn't exactly brimming with features, but it's usable, and it can automatically record your actions for precise, effortless activity recording. All of these options are tied together in the cloud, meaning that you can move seamlessly from one platform to another.

*TimeSlotTracker* is a good program overall, but we have a few misgivings about recommending it. It hasn't been updated in quite a while (not since 2021). It has its users, but the official forum has a fairly low level of traffic. The user interface has a comprehensive multi-pane design with all of the facilities you'd expect from a desktop application, such as drag and drop and pop-up menus. It's a comprehensive desktop Linux time-tracking application.

As it doesn't have a GUI, *Timewarrior* doesn't have the broad appeal of some of the other offerings, but some people will prefer starting and stopping tasks from the command line. There are many system administration scenarios where this could be useful. It is, of course, fully scriptable. If you're prepared to work in a terminal, it can carry out most of the tasks that the other options can, including report and chart generation. There's some opportunity for expansion of what it can do, if you're willing to get your hands dirty with configuration and extensions.

*Arbtt* scratches a different time-tracking itch compared to others because it's a command-line utility that is solely for automatic time tracking. It has quite a lot of scope for customisation in terms of precisely what is being recorded, opening some niche activity monitoring scenarios. However, some of the other options can also do automatic activity recording, and *Arbtt* would struggle to be used as a general-purpose time tracker like the others.

*Hamster* is a simple native Linux utility that carries out basic time-tracking tasks, and it can produce reports. The interface seems positively stripped-down compared to *TimeSlotTracker*. Take a look at it – it might be able to do everything you need if your requirements are simple.



### 1st **TimeCamp** **9/10**

**Web:** [www.timecamp.com](http://www.timecamp.com)

**Licence:** Proprietary **Version:** 1.7.2.0

Generous features in the cloud or the desktop on the free tier.

### 2nd **TimeSlotTracker** **8/10**

**Web:** <https://sourceforge.net/projects/timeslottracker>

**Licence:** GPL v2 **Version:** 1.3.21

Featureful multi-platform time tracker with a good GUI.

### 3rd **Timewarrior** **7/10**

**Web:** <https://timewarrior.net>

**Licence:** MIT licence **Version:** 1.7.1

Command-line-based tracker. Extendable with more advanced features.

### 4th **Arbtt** **6/10**

**Web:** <https://arbtt.nomeata.de>

**Licence:** GPL v2 **Version:** Git (0.12.0.3)

Can automate the gathering of user activity data, but needs configuring.

### 5th **Hamster** **6/10**

**Web:** <https://github.com/projecthamster/hamster>

**Licence:** GPL 3.0 **Version:** 3.0.2-3

Simple GUI time tracker with report output that gets the job done.

## » ALSO CONSIDER

*Toggl* (<https://toggl.com>) is a handy web-based time tracking tool with an extensive free tier. Its Linux standalone client has been deprecated recently, but the service is still usable via the website and mobile app.

*Watson* (<https://tailordev.github.io/Watson>) is a command-line-based tracking app. We would have looked at it here, but we already had two command-line apps on our list.

*KTimeTracker* (<https://apps.kde.org/en-gb/ktimetracker>) is a Linux native time-tracking application, but it had its last major release in 2019, meaning that it's probably not worth getting too invested in learning to use it.

*GTimeLog* (<https://gtimelog.org>) is a GUI program with a command-line feel to it. It's a fairly simple application and it can log time spent on tasks. **LF**



# LINUX INVADES WINDOWS

Join **Nate Drake** as he explores Linux's infiltration of Windows. Have the barriers between operating systems truly been broken?

**I**n late January 2024, Microsoft released an Insider Preview of Windows Server 2025 that seemingly supports the `sudo` command. Naturally, experimental features are just that – currently, the Windows `sudo` can't even be invoked from the command line. Still, if and when it can be, it will simply be the latest chapter in the ongoing story of Linux integration into Windows.

It seems Microsoft's signature tactic of 'embrace, extend, extinguish' is a thing of the past. Instead of developing proprietary versions of open source protocols to muscle the latter out of the market, the tech giant is now one of the biggest contributors to open source, even going so far as to acquire the home of many FOSS projects, GitHub in 2018.

While Windows and Linux remain extremely distinct, both philosophically

and programmatically, in recent years Microsoft has embraced the latter OS to some extent. This includes offering Linux OSES (including its own Sphere) to Azure customers. Windows 10 and 11 users can even use WSL (Windows Subsystem for Linux) to run Linux kernels and apps within the Windows ecosystem.

We're going to delve into the extent to which Microsoft has introduced Linux features into Windows, which includes a discussion of WSL and WSL2. We also explore Microsoft's cross-platform *Visual Studio Code*, which can integrate with WSL2 for testing Linux apps.

Finally, we hearken to the internet's beating toms to tease out clues as to what the future has in store for further Linux integration in Windows.





# Learning to love Linux

The early days of Microsoft put it at loggerheads with the growing open source movement, but things are different now.

**M**icrosoft and its flagship Windows OS have a troubled history with Linux and open source. When Bill Gates penned his 'Open Letter to Hobbyists' in 1976, he complained about the rampant software piracy taking place in early hobbyist communities – particularly that of the Altair Basic interpreter, which he'd co-created. At a time before open source initiatives like the GNU and Linux projects, his opinion was clear: "Are people who copy software any different than those who copy records and books?"

Twenty-two years later, one of the leaked Microsoft Halloween Documents (see box) focused on how the tech giant viewed its rival OS, even showing grudging respect: "Linux represents a best-of-breed Unix, that is trusted in mission critical applications, and – due to its open source code – has a long-term credibility which exceeds many other competitive OSes."

Publicly, of course, Redmond disavowed Linux and open source altogether. In 2001, former Microsoft CEO Steve Ballmer famously said: "Linux is a cancer that attaches itself in an intellectual property sense to everything it touches." Gates even described the open source GPL (GNU Public Licence) as "Pac-Man-like".

Ballmer later backed down from taking such a hard line on Linux. This may have been due to Microsoft's commitment to releasing open source software in the intervening years. This included introducing .NET Core in 2014 – an open source, cross-platform successor to the .NET Framework.

In 2015, Microsoft CEO Satya Nadella even went as far as putting up a slide proclaiming "Microsoft loves Linux". The company has released Linux versions of popular programs like *SQL Server* and *Microsoft Teams*. There's even a dedicated repository wherein users can install select Microsoft tools via *Apt* and *Yum*. Crucially, users can also now make use of WSL2 to run Linux kernels in a lightweight VM.

Today the gods of Redmond are proud members of the Open Source Initiative and have made Linux operating systems available for their Azure Cloud Services. The company joined the Linux Foundation at its top-tier Platinum membership level in 2016.

In 2020, Microsoft president Brad Smith admitted in an interview with ZDNet: "Microsoft was on the wrong side of history when open source exploded at the beginning of the century."

According to the Open Source Contributor Index, Microsoft is now the second largest contributor to open source behind Google, with 4,649 active projects. In October 2023, for the first time, Microsoft went so far as to provide specific steps on its support pages on how to install Linux directly on a PC via dual boot. And since 2009, Microsoft has been contributing to the Linux kernel via the non-profit, Outercurve Foundation.



Altair 8K Basic on paper tape. This was a popular storage medium before floppy disks but it made software very easy to pirate.

However, Windows is still based on the proprietary NT Kernel. Users make use of closed-source apps to manage data on Microsoft's own NTFS filesystem.

Microsoft's Damascene conversion to being a cheerleader of open source could also just be motivated by expectations of Azure customers, who mostly favour Linux and related workflows. After all, while Linux desktop's market share continues to climb, it still is under 4% versus 73% for Windows.

It's also likely that Windows desktop usage will climb even further with WSA (Windows Subsystem for Android), which offers users a way to run Android apps from within Windows. What was traditionally a perk of Chromebooks may soon become commonplace on all Windows devices.

Once again, this raises the spectre of whether what is good for Microsoft is necessarily good for Linux and open source. The best way to answer these questions is through careful understanding of exactly how far Linux has encroached into the Windows desktop.

## QUICK TIP

If you want to use WSL2, you need a machine running Windows 11 or Windows 10 version 1903 or higher, with Build 18362 or higher. Because WSL2 uses virtualisation, it doesn't function properly inside a virtual machine.

## » THE HALLOWEEN DOCUMENTS

A long-standing source of understandable anti-Microsoft sentiment from the open source and Linux community was the leaked 1998 Halloween Documents. These were a series of internal confidential Microsoft memos that detailed the threat to its business from open source in general and specifically Linux. They can be read at [www.catb.org/~esr/halloween](http://www.catb.org/~esr/halloween). They're ancient history but the opening summary shows how seriously Microsoft took Linux as a threat:

"Linux poses a significant near-term revenue threat to Windows NT Server in the commodity file, print and network services businesses. Linux's emphasis on serving the hacker and Unix community alleviates the near-medium term potential for damage to the Windows client desktop. In the worst case, Linux provides a mechanism for server OEMs to provide integrated, task-specific products, completely bypassing Microsoft revenues in this space."





# Linux inside Windows

Run Linux apps on Microsoft's OS in Windows Subsystem for Linux 2.

**T**he origins of WSL (Windows Subsystem for Linux) lies in Microsoft's Project Astoria, a tool originally designed to port Android apps to Windows 10 Mobile. The project was killed in 2016 but the code formed the basis of WSL, which was released in beta later in the same year. Unlike third-party apps like Cygwin, WSL's initial aim was native Linux compatibility through serving Unix programs as custom isolated 'pico processes'.

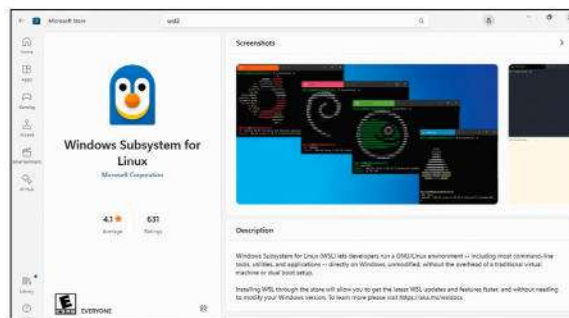
In May 2019, Microsoft announced the release of WSL2, a huge improvement over its predecessor, given it uses virtualisation to run Linux kernels inside a lightweight virtual machine. This not only improves performance but adds full system call compatibility.

Both WSL2 and compatible Linux distros can easily be installed via the command line or Microsoft Store.

## WSL's 'so what?' moment

While WSL2 is certainly a much more efficient way of implementing Linux in Windows, it still needs to pass the 'so what?' test. In other words, why shouldn't Linux developers and other interested parties just keep on running full distros of Linux in a dedicated virtual machine using software like *VMWare* or *VirtualBox*?

The fact, for instance, that WSL2 now has GPU support and can even run GUI Linux apps is hardly



WSL2 can be installed via `cmd.exe` or the Microsoft Store. You can also use the store to install more Linux distros.

enough reason to switch, as most VM software can do the same. As far as developers are concerned though, there's every reason to take WSL2 for a spin. Not only is the setup incredibly simple but you can easily create and switch between distros with just a few keystrokes.

Files can be stored in isolated Linux filesystems, and updates can be carried out automatically across every installed distro. Command-line tools such as *Bash*, *Vim* and so on are supported out of the box and you can install more via the distro's own package manager.

Crucially, using WSL2 gives you the ability to invoke Windows applications via the shell. Installed Linux apps

## » TERMINAL CASE

By default, WSL2 launches in *Windows Terminal*. While we don't make a habit of recommending Microsoft products at **LXF**, this is one that integrates nicely with installed Linux distros.

If you're serious about using Linux on a regular basis, launch *Windows Terminal* via the search bar, then click the arrow next to the new tab icon (+).

From here you can switch between profiles – for example, from *PowerShell* to *Ubuntu*. To switch to the Linux command line by default, hit *Settings*. In the new window, select the drop-down menu in the *Default Profile* section and change to your chosen distro.

From here you can also change your default terminal



You can use *Windows Terminal* and third-party utilities such as *ConEmu* (pictured) to automatically launch your chosen Linux distro via WSL2.

application from *Let Windows Decide* to *Windows Terminal*. We also recommend enabling *Launch On Machine Startup*.

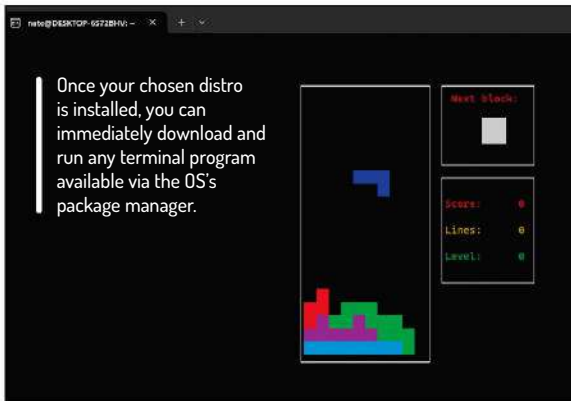
If you want to use the Linux command line across multiple distros and/or have a preferred program, WSL2 is compatible with other

terminal utilities, such as *Hyper* or *ConEmu*.

By default, these programs launch `cmd.exe` but you can configure your profile to point to `%windir%\system32\wsl.exe` instead. The steps to do this differ depending on your terminal program. For instance, on first launch *ConEmu* displays a fast configuration screen. In the *Choose Startup Task* field, you only need change `{Shells:cmd}` to `{Shells:wsl}`. You can also go to *Settings > Tasks* to define parameters.

Regardless of the terminal program, you can find help with basic WSL commands, including running specific apps in installed distros, at <https://learn.microsoft.com/en-us/windows/wsl/basic-commands>.





are also listed in the Windows Start Menu, so can easily be launched there or via the command line.

### Start your subsystems!

If you're now convinced it's worth taking WSL2 for a spin, first make sure that your copy of Windows is up to date by typing **updates** into the search bar and choosing Windows Update. If you're using Windows 190, you need to be running version 21H1, 21H2, or 22H2 in order to use WSL2.

WSL2 also uses Hyper-V. This requires 64-bit hardware virtualisation support which you must enable in your BIOS. Check with your motherboard manufacturer if you're unsure how to do this.

Once this is done, save and close any open files. Command-line lovers can run *cmd.exe* or open Windows PowerShell to begin installation with:

```
$ wsl --install
```

If the installer hangs (as it did for us), or you're more a fan of the GUI, you can also search for WSL2 in the Microsoft Store and hit Get to begin setup.

Once the install is complete, you need to check that Virtual Platforms are enabled. In the Windows search bar, find and select Turn Windows Features On Or Off. From here you can check the box marked Virtual Machine Platform.

By default, WSL2 installs Ubuntu but you can install another distro if you wish. If you used the Microsoft Store to search for WSL2, you can see all available distros listed and install them by clicking Get.

Still, the best way to view currently available distros is via the command line: **c**.

Install your chosen distro using:

```
$ wsl --install -d <Distribution Name>
```

For example:

```
$ wsl --install -d <kali-linux>
```

For this tutorial, we're using Ubuntu 22.04 LTS. If you want to install a distro that isn't available for download, WSL2 also supports importing other distros via a tar file. Follow the steps on Microsoft's Support site to do this (<https://learn.microsoft.com/en-us/windows/wsl/use-custom-distro>).

Once install is complete, restart your PC. After logging back in to the desktop, open the Start Menu to view a shortcut to your Linux distro of choice. If you click to launch, your distro opens in *PowerShell*. You're prompted to set a username and password.

Next, make sure your OS is fully up to date with:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

If you're using *PowerShell*, you can also check for kernel updates with:

```
$ wsl --update
```

Installing and running programs now works in the same way as any terminal utility in Linux. For instance, try installing popular *Tetris* clone *Bastet* with:

```
$ sudo apt-get install bastet
```

### Introducing GUI apps

Installing terminal utilities and games is all well and good, but frankly you can achieve much the same thing by logging into a remote Linux server via SSH.

The real fun lies in the fact that Windows now supports both X11 and Wayland Linux GUI applications. This means you can launch Linux apps from the Start Menu, pin them to the Windows taskbar, switch between them and Windows apps using tabs, and even use copy and paste.

To get started, you need to install a driver to use a virtual GPU (vGPU) that supports hardware-accelerated OpenGL rendering. You can find links to drivers for popular graphics cards at: <https://learn.microsoft.com/en-us/windows/wsl/tutorials/gui-apps>.

You also need to have at least Windows 10 Build 19044+ or Windows 11 installed to use this feature.

Once you have downloaded and installed the necessary GPU installers, open *PowerShell* and then update it with:

```
$ wsl --update
```

Next, you need to restart WSL2 for this update to take effect by running:

```
$ wsl --shutdown
```

Use the Start Menu to launch your chosen distro again, then update the OS itself again with:

```
$ sudo apt-get update
```

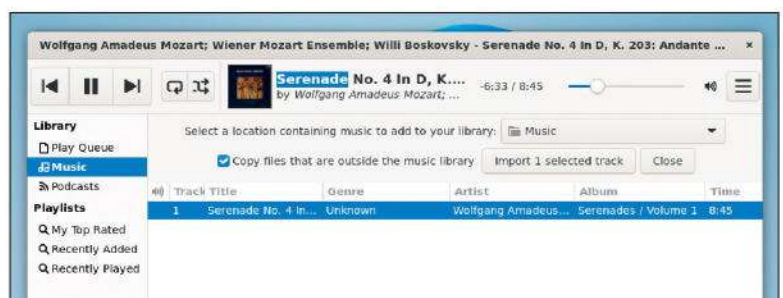
You can now install a GUI app, such as the *Rhythmbox* music player, for example, which is usually only available for Linux:

```
$ sudo apt-get install rhythmbox
```

Once the install is complete, you can launch the app in question by searching via the Windows Start Menu. If you did choose *Rhythmbox* or any app from which you need to access local files, this is a good time to learn how they're managed in WSL2.

Windows files can be accessed from within your Linux distribution from **/mnt/**. For instance, we pointed *Rhythmbox* towards our music folder at **/mnt/c/Users/nate1/Music**.

You can equally use the Windows File Explorer to browse your Linux directories. Simply use **cd** to navigate to your chosen parent folder via the command line and then run *explorer.exe*.



Linux apps can access Windows files and folders via **/mnt**. Here, *Rhythmbox* is playing an MP3 file in the Windows Music folder.



# VS Studio for Linux



WSL2 integration offers the best of cross-platform development.

**M**uch as we're loath to admit it as Linux lovers, when it comes to development, you can do much worse than Microsoft's own *Visual Studio*.

For almost 30 years, this robust IDE (integrated development environment) has been used to develop everything from basic scripts to websites and apps.

*Visual Studio IDE* is soon to be Windows only (*VS Studio for Mac* is due to be retired in August 2024). However, the standalone source code editor *Visual Studio Code* is available for Windows, Mac OS and Linux. While much more lightweight than its heavier IDE cousin, *VS Code*'s functionality and supported languages can be enhanced via a number of extensions.

Most importantly, there's an extension to connect *VS Code* with WSL2. This means you can develop apps on Windows while also running them in a dedicated Linux kernel. This makes it much easier to ensure cross-platform compatibility for your apps. It's also hugely more efficient than other options, such as trying to mirror code across both physical and virtual machines.

## Starting the Studio

Before integrating WSL2 with *VS Code*, we recommend repeating the steps on the previous pages to set up a dedicated Linux instance for running your chosen apps.

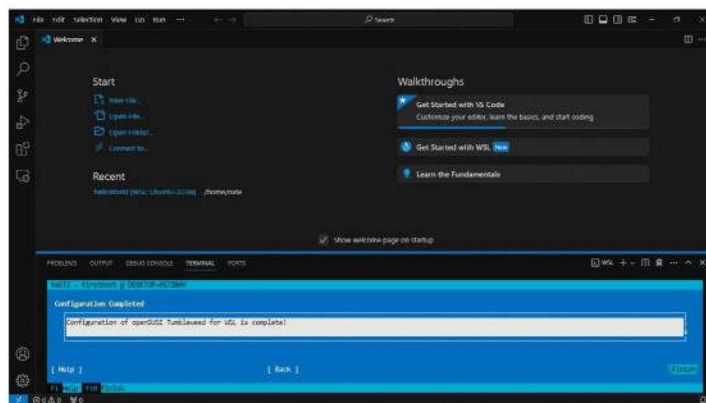
Next, you need to install *VS Code* via the Microsoft website: <https://code.visualstudio.com/download>.

The installer is available for Windows 10 and 11. By default, the download icon accesses the User Installer, which installs to your **User** folder, so doesn't require admin privileges. A System installer is also available.

The installer is straightforward and offers numerous extra options. Choose freely among the rest, such as creating a dedicated Start Menu icon, but make sure the option Add to PATH is checked.

Once install is complete, open *VS Code* via your new desktop shortcut or the Start Menu. Look to the left-hand pane and click the extensions icon. In the search box that appears, type `wsl`. Click Install to continue.

Sharp-eyed users will no doubt have noticed a new status bar item, bottom-left. This is the Remote Status



You use the Command Palette in *VS Code* to set up and install more Linux distros.

section, which you use to check if you're running *VS Code* in a 'local' (Windows) or 'remote' (WSL) context.

## Prod the python

Next use the Start Menu to launch your chosen Linux distro, or open *Windows Terminal* and run `wsl`.

For the purposes of this guide, we're going to create a simple script in Python. Unlike most Unix systems, Windows doesn't come with the language installed by default. As far as we can tell, all the Linux distros offered for install via WSL2 via the Microsoft Store do have Python 3 preinstalled. You can double-check this in *Windows Terminal* by running:

```
$ python3 --version
```

If Python 3 isn't installed, simply run:

```
$ sudo apt-get update
```

```
$ sudo apt-get install python3 python3-pip
```

Make a folder for testing purposes and switch to it:

```
$ mkdir helloWorld
```

```
$ cd helloWorld
```

Next, create a simple test script:

```
$ echo 'print("Hi, don't forget to subscribe to Linux Format!")' >> hello.py
```

Test it by running locally from the Linux instance:

```
$ python3 hello.py
```

## Linux coding in Windows

At this stage, you have created your own Python script within what amounts to a remote Linux environment. It

The WSL extension uses a client-server architecture. The client runs 'locally' in Windows and the server 'remotely' in your Linux distro.

```
nate@DESKTOP-6S72BHV: ~/helloWorld$ python3 hello.py
Hi, dont forget to subscribe to Linux Format!
nate@DESKTOP-6S72BHV:~/helloWorld$ code .
Installing VS Code Server for x64 (903b1e9d8990623e3d7da1df3d33db3e42d80eda)
Downloading: 100%
Unpacking: 100%
Unpacked 1530 files and folders to /home/nate/.vscode-server/bin/903b1e9d8990623e3d7da1df3d33db3e42d80eda.
nate@DESKTOP-6S72BHV:~/helloWorld$
```

can be run from within the distro. You can even edit it using *Nano* or *Vim*.

Still, if you're intent on developing a cross-platform app, remember that as things stand, you can't meaningfully debug and run apps from Windows itself without installing the same Python development stack. This is fairly simple to do but undermines the advantage of having your own lightweight, standalone Linux instance with all necessary Python tools preinstalled.

Fortunately, you can use WSL2 to edit, debug and run your script from VS Code. To get started, double-check the terminal to make sure you're still in the same folder as your test script, then run `code .` – the `.` tells VS Code to open in the current folder.

The terminal now displays a message stating **Installing VS Code Server**. You'll also see a number after this – this is the version, which automatically matches that of the VS Code client you installed directly in Windows earlier.

In simplest terms, in running this command, you're having VS Code install a small server to which the VS Code client on Windows itself can connect.

The VS Code server also installs host extensions in WSL, so that they run in the context of the tools and frameworks installed therein. This is independent of the language extensions and frameworks that are installed in Windows, which is the optimal setup for a development environment.

Once install is complete, you'll see a notification that VS Code is connecting to WSL. You may also be prompted to authorise access to the Node.js-based server. The program itself then starts.

The first pop-up window asks you to confirm that you trust all files in the parent folder – in the case of this test, this is your **home** directory. Click Yes, I Trust The Authors to continue.

You can now check the Remote Status section at the bottom-left to confirm that VS Code is accessing your chosen WSL instance.

For further confirmation, take a moment to hover your mouse over the test script **hello.py** and note the file path. You'll see that it matches the path in your chosen Linux distro.

Click on **hello.py** to begin editing. As this is a Python script, in this case VS Code prompts you to install the Microsoft Python extension for rich editing and improved debugging. Click Install to proceed.

Once this is complete, take a moment to click the extensions icon in the left-hand pane, or list them using `Ctrl+Shift+X`. From here you can see the 'locally' installed WSL extension, as well as a new section entitled WSL: [OS Name] – Installed. This shows that the extension is installed on the WSL side.

While you're here, you can also search for and install other extensions into your WSL instance via VS Code, such as the Pylint linter. This provides a richer palette as well as completion for most commands, making debugging much easier.

Next, take your code for a test spin by hitting `F5`, then choosing Python Debugger from the drop-down.

## Bugging out

Given that our test script only constitutes one line, it doesn't offer many opportunities to allow you to step through the code or show off VS Code's debugging

## » THE COMMAND PALETTE

You can access more VS Code WSL commands by using the shortcut `Ctrl+Shift+P` to bring up the command palette.

Type `ws` to list options. One of the most useful is Connect To WSL Using Distro. If you select this, VS Code displays a list of all installed Linux distros. You can also click Add New to list more and launch the installer from VS Code. This must be done locally – you must have closed the 'remote' WSL connection from VS Code first.

If you then run Connect To WSL Using Distro once again and choose the newly installed OS, VS Code also installs the necessary server software on your target system so it can connect.

If your program is stored in a specific place, save the trouble of navigating to it via *Windows Terminal* by using Open Folder In WSL. If you're connected to an WSL instance, you can navigate from here to the directory containing your code. As before, click Show Local to display the 'local' Windows filesystem. If this is too much trouble, the Command Palette also contains a Reopen Folder In Windows option.

We recommend the Help command that links to Visual Studio's documentation on WSL (<https://code.visualstudio.com/docs/remote/wsl>). From here you can learn more about how the WSL extension works in VS Code, such as the fact it can run other extensions and commands directly. This is also where we learned that you can open an WSL window directly from VS Code: just press `F1`, then select WSL: Connect to WSL to launch your default distro.

You may want to follow the link to install the Remote Development Extension Pack with its extensions for WSL, SSH and containers.

features. Still, you can engineer a crisis yourself by clicking on line 1 and then pressing `F9` to create a breakpoint.

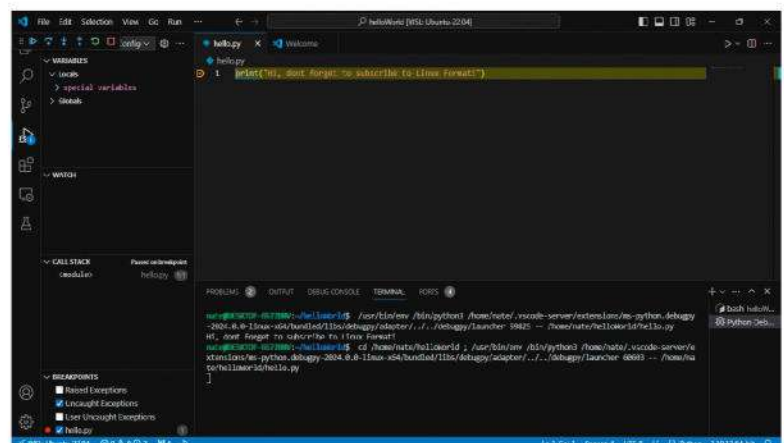
Try to run the code again using `F5`, then select Python File. Your code now starts and hits the breakpoint. You'll see you can now inspect variables, create watches and navigate the call stack. You can also press `F10` to see the output of the print statement in the debug console.

To access other scripts stored in your WSL instance from VS Code, simply use `File > Open`. You'll notice the directory tree here is for your Linux instance, though as before, you can still access Windows files via `/mnt` – for example, `/mnt/c/`.

If you do want to switch back to Windows files only, simply click the button marked Show Local. This launches the standard Windows Open dialog.

To end your WSL session and return to running code locally, select `File > Close Remote Connection`.

Add a breakpoint to your test script in VS Code to inspect variables, create watches, and navigate the call stack.





# Run the Microsoft distro

Rumours are rife but how far will Redmond go to embrace Linux...



**D**espite its easy integration with WSL, in October 2021 *Visual Studio Code* became the site of a battleground between Microsoft and the open source community. The controversy centred around the release of .NET 6. The tech giant had originally hinted it would include a feature to allow developers to modify source code while an app is running and immediately see the results.

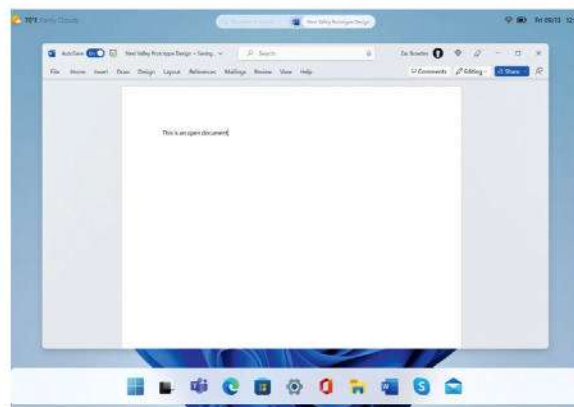
Microsoft provoked a backlash due to its last-minute decision to include this feature in *Visual Studio* only – a paid product primarily designed to run on Windows. Ultimately, Microsoft bowed to pressure from both the open source community and its own employees to grant a pull request to enable this feature with VS Code, but events like these beg the question: can a commercial company like Microsoft ever fully embrace open source?

## Microsoft opens up

During the late '90s, Microsoft released SFU (Services For Unix), which included some GNU utilities. Given that these were covered by the GPL, Microsoft lived up to its obligations and offered selected components under the same open source licence.

It wasn't until 2004 that Microsoft released a product as open source of its own volition: WiX (Windows Installer XML). The licence used was the Common Public Licence, later superseded by the MS-RL (Microsoft Reciprocal Licence). This licence has received the approval of the Open Source Initiative but the Free Software Foundation does not consider it to be compatible with the GNU GPL.

The FSF feels much the same about the Microsoft Public Licence, which has been used to release other open source projects, such as the Windows Template Library. Terms like “permissive” and “weak copyleft” are often bandied around by FOSS advocates, but these Microsoft licences do allow creation of derivative works using open source code but restrict the use of Microsoft-specific names and trademarks.



A preview of the next version of Windows leaked from Ignite 2022. Will it really run Linux under the hood?

In the intervening years, the tech giant seems to have learned its lesson by releasing *PowerShell Core* under the GNU-compatible MIT Licence in 2016. As this licence is permissive, it can be shipped with apps. In theory, this means that any scripts run in the target environment run on the same shell using the same command implementations as the test environment.

Given that Unix systems already have strong shell environments that can accomplish much the same thing, this raises the same issue as with WSL and VS Code: is Microsoft truly embracing open source, or simply providing a way for developers to stay within the Windows ecosystem?

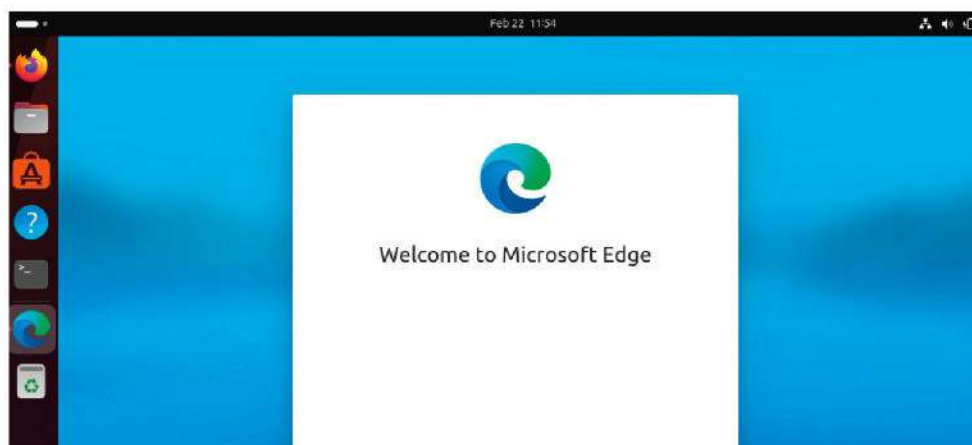
Still, it's harder to be cynical about other major moves by Microsoft. In 2019 alone, the company released *Windows Terminal*, *PowerToys* and the Microsoft C++ Standard Library as open source.

In the same year, Microsoft developers also began work on transitioning Windows' *Edge* browser to use the open source *Chromium* browser as a codebase. This was a bold move and involved almost completely abandoning the version of *Edge* developed in-house. Speaking in an interview with *The Verge*, Microsoft

spokesperson Joe Belfiore admitted: “Our volume use is low, partly because we're only on Windows 10.” This reflected the fact that at the time *Edge* only came bundled with the latest version of Windows, leaving many Windows 7 users in the dust. Even those willing to make the upgrade found that browser updates were usually only released along with Windows 10 feature updates around twice a year.

Work began on a prototype code-named *Septagon*, based on *Chromium*. After getting the green

The Linux version of Edge running in Ubuntu 23.10. Installation is available via Microsoft's repositories or standalone DEB/RPM files.



light from Microsoft top brass, such as Bill Gates, Microsoft began working with Google engineers to develop the modern *Edge*, which now is the second-most popular web browser after *Chrome*.

Although the decision was pragmatic, it's a win for open source and collaboration. Microsoft clearly felt its open source bona fides were strong enough to woo Linux users, because after previews, a stable version of *Edge* was made available to Linux users in 2021.

Despite being installable via the Microsoft repo and enjoying compatibility with most *Chrome* extensions, few Linux users headed for *Edge*, and *Firefox* and related browsers remain most popular on the OS.

The wooing has since continued with successive versions of WSL and now WSA to allow running Android apps on Windows devices.

### From pirates to mariners

WSL2's kernel was originally developed by Microsoft's Linux System Group. Since 2020, the same team has also been maintaining CBL-Mariner. CBL (Common Base Linux) is a fully-fledged Linux distro, albeit one that only comes with the basic packages needed to support and run containers (much like Fedora CoreOS).

It uses an RPM-based package manager and RPM-OSTree for system updates and rollbacks. The Linux System Group also clearly takes security seriously, as Mariner comes with features like a hardened kernel, signed updates and tamper-resistant logs. Crucially, Microsoft doesn't seem to be pulling any punches any more when it comes to open source: the entire OS is covered under a mix of MIT, Apache and GPL licences.

Official ISO builds can now be downloaded via Microsoft's servers. While technically it can boot on any x86\_64 PC, it's primarily designed to work on a hypervisor platform, such as Microsoft's own *Hyper-V* or *VirtualBox*. For the purposes of this guide, we've chosen *VirtualBox*, given it can run on both Windows

and Linux. To get started, download the CBL-Mariner ISO from [https://aka.ms/mariner-2.0-x86\\_64-iso](https://aka.ms/mariner-2.0-x86_64-iso).

You can also compile the ISO yourself from the Linux terminal if you prefer (see box, below).

Once the download is complete, open *VirtualBox* and select Add. Set a name for the VM, such as *Mariner1*, then under ISO Image, navigate to the file you downloaded. Under Type choose Linux, then select Other Linux (64-Bit) for Version.

The default settings of 512MB RAM and 1 CPU are sufficient for taking Mariner for a test drive but feel free to adjust. Hit Next to create a virtual hard disk (we recommend a minimum of 15GB). Click Next > Finish.

Click Start to launch the VM. On first boot, choose the graphical installer. You are given the choice to install the core or full version. Choose freely, then click Next. Agree to the T&Cs, then click Next once again to be taken to the Partitions screen. Note that Mariner supports hard disk encryption. Choose Next again to create a user account with a suitably strong password, then Install > Install Now to proceed.

Once install is complete, hit Restart to boot the machine. *VirtualBox* should automatically eject the ISO. You can now sign using the credentials you created.

CBL-Mariner uses *Systemd* for managing all running services, so for your first command, run `systemctl` to view descriptions of current loaded and active ones.

### Microsoft loves Linux

Twenty-five years ago, it would be hard to imagine that Microsoft would be a member of the Linux Foundation, not to mention mirroring Linux devs by supporting rewrites of the kernel in Rust and releasing open source distros. Naturally, innovations such as WSL2 and CBL-Mariner could just be there to keep programmers using Windows and Azure. Open source could just be a form of 'Windows dressing'. Still, Linux is in the OS – and it's here to stay. **LXF**



## » BUILD YOUR OWN CBL-MARINER ISO

Microsoft is now offering precompiled ISOs of CBL-Mariner. This saves the trouble of compiling either your own ISO or bootable VHDX/VHD image based on the project's source code.

Still, if you prefer a DIY approach, you can compile your own CBL-Mariner ISO in Linux. Open the terminal in your chosen distro and run:

```
sudo apt-get install build-essential git go
```

Next, you need to use Git in order to clone the CBL-Mariner repository:

```
git clone https://github.com/microsoft/CBL-Mariner.git
```

Once the repository has been cloned to your device, switch to the CBL-Mariner

**toolkit** directory:

```
cd CBL-Mariner/toolkit
```

You can now execute the ISO build command with:

```
sudo make iso REBUILD_TOOLS=y REBUILD_PACKAGES=n CONFIG_FILE=./imageconfigs/full.json
```

Once it's complete, you'll find your ISO in **CBL-Mariner/toolkit/out/images/full**.

If you would rather build your own VHDX image, run:

```
sudo make image CONFIG_FILE=./imageconfigs/demo_vhdx.json
```

If you prefer a VHD image, run the above command, but substitute `vhd` for `vhdx`.

To check the packages that go into your chosen

```
nate@ubuntu-12: ~/CBL-Mariner/toolkit
nate@ubuntu-12:~/CBL-Mariner/toolkit$ sudo make iso REBUILD_TOOLS=y REBUILD_PACKAGES=n CONFIG_FILE=./imageconfigs/full.json
SUDO USER is set, running 'go get' as user 'nate'
go: downloading golang.org/x/vuln v0.14.0
go: downloading golang.org/x/exp v0.0.0-20200219135020-9ba8301b415c
go: downloading golang.org/x/mod v0.4.0
go: downloading golang.org/x/sys v0.0.0-20200219135020-9ba8301b415c
go: downloading golang.org/x/text v0.3.2
go: downloading golang.org/x/tools v0.0.0-20200219135020-9ba8301b415c
go: downloading golang.org/x/xerrors v0.0.0-20200219135020-9ba8301b415c
go: downloading github.com/alecghom/kingpin v2.2.6
go: downloading github.com/sirupsen/logrus v1.9.3
go: downloading github.com/bendahl/uintput v1.4.0
go: downloading github.com/gdamore/tcell v1.4.0
go: downloading github.com/asaskevich/govalidator v0.0.0-20210307081110-f21760c49a8d
go: downloading gopkg.in/yaml.v3 v3.0.1
go: downloading github.com/rivo/tview v0.0.0-20200219135020-9ba8301b415c
go: downloading github.com/muesli/crunchy v0.4.0
go: downloading github.com/cavaliercoder/go-cpio v0.0.0-20180626203310-925f9528c45e
go: downloading github.com/klauspost/pgzip v1.2.5
```

Microsoft now offers downloads of CBL-Mariner 2.0 ISOs but you can compile them yourself via the Linux terminal if you prefer.

image, you can view the full list at <https://packages.microsoft.com/cbl-mariner/2.0/prod/>.

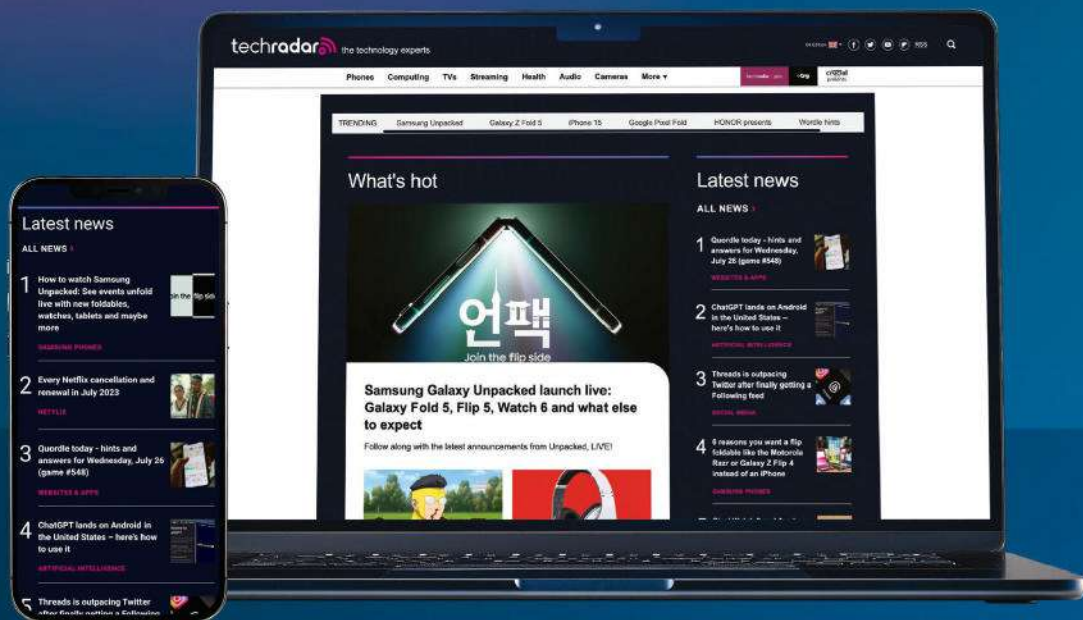
As you'll see, these are fairly spartan compared to the likes of Ubuntu Server, but exactly what you need

for container images running on Azure.

Alongside Microsoft's own tools, such as .NET and ASP.NET Core, the CBL-Mariner repository contains common Linux tools including Python 3, Kubernetes and Node.js.



# Meet the technology experts

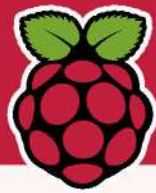


- The world's **most** comprehensive technology website
- An **unrivalled** mix of news, opinions, reviews and features
- **All-new design**, new homepage, new features and special reports
- Backed by **over 300 years** of editorial experience

**techradar**  
the technology experts

**www.techradar.com**





## Turning the Pi Pico into a custom GPU

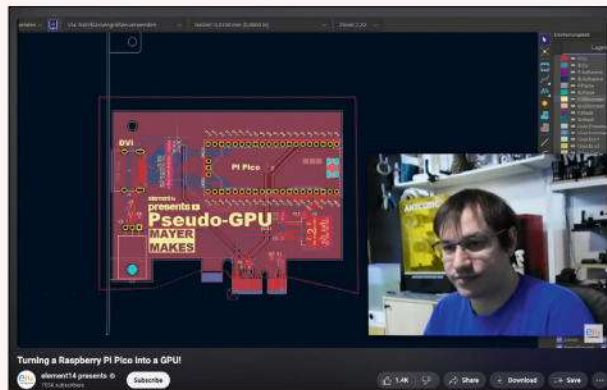
Renowned fanatical Pi modder Clem Mayer goes all out and creates a Pi-powered graphics card.

**W**hen the Pi Pico was first launched, there was talk about using its I2C capabilities to run DVI outputs, which is to say to drive a digital display. There already exists a Pi Pico DVI library called PicoDVI (<https://github.com/Wren6991/PicoDVI>) that enables a Pico to control a digital display over HDMI.

Utilising this, Clem Mayer, after initially running into timing issues when attempting to directly connect the HDMI to the Pico, found more success once he used a Pico DVI Sock (<https://github.com/Wren6991/Pico-DVI-Sock>), which is a small PCB that sits under the Pico with an HDMI mounted on it.

Mayer successfully initiated the new pseudo Pico GPU with

a Hello World display. To turn his creation into a true GPU, he devised a PCIe PC expansion card to mount the Pico to and host the HDMI connector, which while pushing the definition a bit, is a Pico-powered PC graphics card. See [www.youtube.com/watch?v=8-RuYM-9s3A](https://www.youtube.com/watch?v=8-RuYM-9s3A)



Behold the world's first Pi Pico GPU expansion card. No, it can't play Crysis.



**Les Pounder** works with groups such as the Raspberry Pi Foundation to help boost people's maker skills.

### » RASPBERRY PI TURNS TWELVE!

As I write this, we have just celebrated 12 years (or three if you follow leap years) of Raspberry Pi. It is hard to think of a time when there wasn't a low-cost, credit-card-sized computer for our projects. In those 12 years, Raspberry Pi has sold 61 million units ([www.tomshardware.com/raspberry-pi/raspberry-pi-celebrates-12-years-as-sales-break-61-million-units](https://www.tomshardware.com/raspberry-pi/raspberry-pi-celebrates-12-years-as-sales-break-61-million-units)). Of that total, 57 million are Linux-based computers, the remaining four million being Raspberry Pi Pico microcontrollers.

Let that sink in: 61 million units is a lot of single-board computers or microcontrollers powering hacks as diverse as screaming jelly babies (the original GPIO Hello World back in 2012) to powering industrial machinery. The Raspberry Pi has become a major part of the Linux ecosystem and it has inspired and educated many of those who now work in open source and Linux projects.

Back in 2012, we didn't know what to expect from the Raspberry Pi. Initially it was a single-core Arm CPU running at 700MHz, with only 256MB of RAM. A later revision saw double the RAM but it wasn't until the Raspberry Pi 2 that we saw a quad-core Arm CPU.

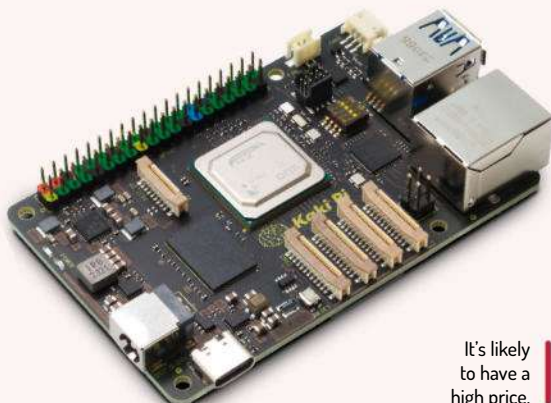
Fast-forward to 2024 and we have the potent Raspberry Pi 5. Quad-core Arm CPUs running at 2.4GHz, easily overclocked to 3GHz, 4 or 8GB of RAM, and now we have easy access to NVMe. I'm not going to say that 2024 is the year of the Linux desktop, but we now have a low-cost Linux desktop that can also be used in many different projects.

Happy birthday, Raspberry Pi!

CREDIT: Element14 Presents, YouTube

## Kaki Pi From Japan!

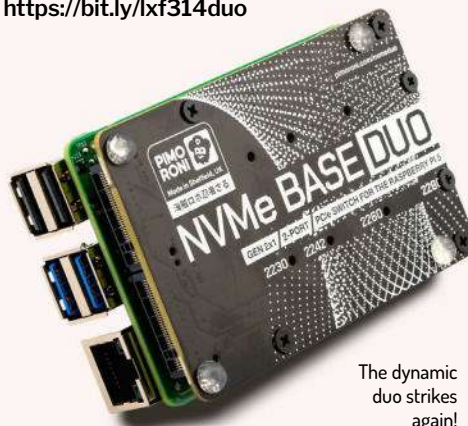
Another Pi-like SBPC appears! From Yuridenk-Shokai in Japan, this packs a Renesas RZ/V2H Cortex-A55/R8/M33 MPU with 80 TOPS of AI. It's aimed at robotics and has four CSI video inputs along with a PCIe 3 connection. Find out more in Japanese: [www.kaki-pi.ai](https://www.kaki-pi.ai)



It's likely to have a high price.

## Double the fun NVMe on Pi 5

We were all getting excited about having one NVMe drive on our Pi 5s, but now Pimoroni has a dual-NVMe model called the NVMe Duo Base. Importantly, it follows the new PIP design standards and supports all sizes of NVMe drives. Find out more: <https://bit.ly/lxf314duo>



The dynamic duo strikes again!

CREDIT: SYuridenk-Shokai/Pimoroni

# ParrotOS

**Les Pounder** has watched Hackers at least 12 times and knows what all the books are. Now he just needs a cool haircut and RISC-based laptop.

## IN BRIEF

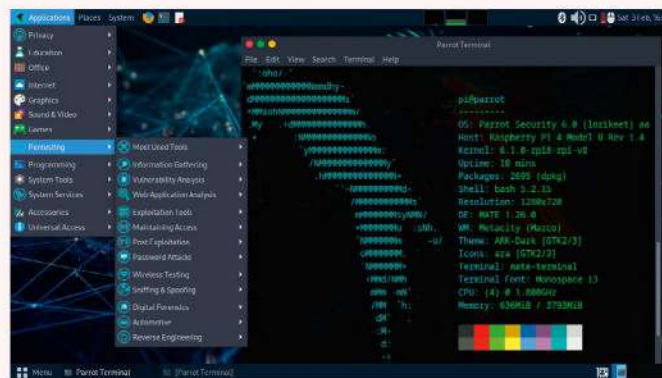
A security focused operating system that is well suited to the Raspberry Pi 4 2GB and upwards, but not Pi 5 for now. ParrotOS provides a plethora of security auditing tools for both red and blue team members. This is a great distro for learners and networking experts to use in their job.

**P**arrotOS is an interesting choice for the Raspberry Pi. We generally think of the Pi as a learning tool for coding, building projects and advancing our computing knowledge. Another aspect of learning with the Pi is security. Whether you are on the red or blue team, you need a toolkit, and within that will be a Raspberry Pi. ParrotOS is an alternative to Kali Linux, and we can debate the pros and cons of both until the cows come home, but let's uncage this bird and see how it flies.

ParrotOS comes in a number of flavours; there are versions for laptop or desktop and the Raspberry Pi. For each architecture, there is a Core (no desktop), Home (lightweight, aimed at home users or developers) and Security Edition (which contains all the 'hacking' tools you could think of). We tested the Security Edition with a Raspberry Pi 4 4GB – right now, ParrotOS doesn't work on the Pi 5.

On first boot, after the drive resizes, we are left at a login. Enter the default password and we are greeted with the Mate desktop, which provides a light feel, even on the Raspberry Pi 4. ParrotOS is based on Debian Bookworm and that means we have a strong foundation to work from. All the usual apps are there: browsers, office, graphics, coding and so on. But there is another section: Pentesting. Penetration testing is a security exercise that involves security experts attempting to penetrate a company's machines. With the correct authority and skills, this is a business model that many companies use to keep their systems secure. All you need to learn is a Pi and your own network to play inside.

Inside the Pentesting menu, we find submenus for Information Gathering, Vulnerability Analysis, Password Attacks and Wireless Testing. If *Wireshark*, *Aircrack* and *Nmap* mean anything to you, you know that ParrotOS has the tools for you. We also spotted a submenu for NFC and RFID analysis; interesting stuff if you have the correct components to make it work. Our only gripe with the Pentesting menu is that many applications require



The main menu hides much more under Pentesting. Here we have plenty of tools to monitor and audit different cybersecurity scenarios.

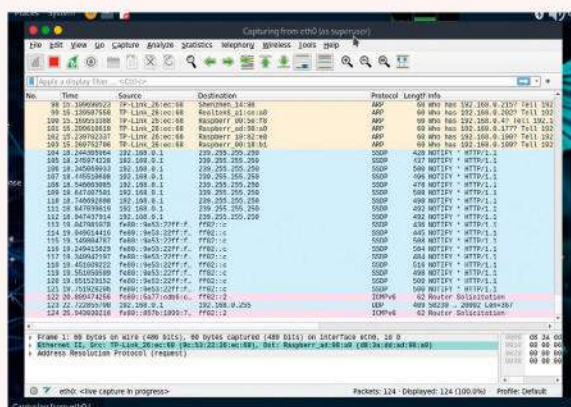
running as root, so selecting their entry from the menu often results in an error. Run the same command from the terminal as root and you are ready to work.

We ran *Wireshark* on our test network and could see traffic between our desktop and wireless printer, then our phone and the Pi itself. *Wireshark* is a powerful tool if used correctly – and legally, of course.

As for the more mundane Pi-centric features, yes we have the GPIO at our disposal. Both GPIO Zero and RPi.GPIO worked, no problem. HAT compatibility is an issue due to PEP668 being used in Bookworm. You need to use a virtualenv in order to install the HAT software, and some just don't like that. We tested Pimoroni's Explorer HAT Pro and it failed to run. Wi-Fi and Bluetooth worked well, as did Ethernet, but that is a given really.

ParrotOS is great for learning pentesting and 'hacker' skills. Yes, it can be used for nefarious and illegal means, but that is not the intent of its creators. ParrotOS is a means to teach learners the skills of their chosen career using cheap hardware that is already in the toolkit of many hackers. We know of one security expert who hid the original Pi in a VOIP phone as part of a security audit.

If you are looking to learn pentesting and security skills, ParrotOS could be for you. Just use it wisely. **UX**



Monitoring network traffic with *Wireshark* is sheer simplicity. You can see exactly what is going on across all your devices.

## VERDICT

**DEVELOPER:** Parrot Security  
**WEB:** <https://parrotsec.org>  
**LICENCE:** Mixed

<b>FEATURES</b>	<b>9/10</b>	<b>EASE OF USE</b>	<b>8/10</b>
<b>PERFORMANCE</b>	<b>9/10</b>	<b>DOCUMENTATION</b>	<b>9/10</b>

Provides every security-related tool we could legally ever need. A great start for those wishing to learn new skills.

» **Rating 9/10**



# Bambu Lab X1-Carbon

There's a reason this Bambu is shooting up all over and **Andrew Sink** is happy to help it out.

## SPECS

**Size:** 389x389x590mm  
**Build:** 256x256x256mm  
**Type:** 1.75mm PLA, PETG, TPU, ABS, ASA, PVA, PET  
**Levelling:** LIDAR/force gauge automatic mesh  
**Nozzle:** 0.4mm (0.6mm, 0.8mm, 0.2mm optional), 300°C  
**Bed:** 120°C  
**Speed:** 500mm/s max  
**Bed:** Magnetic flexible removable platform, textured PEI optional  
**Comms:** MicroSD, Wi-Fi  
**Control:** 5-inch colour touchscreen LCD  
**PSU:** 350W

**O**riginally launched as a Kickstarter campaign in May 2022, the Bambu Lab X1-Carbon with AMS has gone on to become an industry standard in the prosumer market. Capable of extruding high temperature materials, printing with multiple colours simultaneously, and offering advanced print controls, the X1-Carbon has a list of features that is rapidly changing the landscape of desktop 3D printers. LIDAR-assisted bed levelling, automatic flow calibration, and a webcam that can automatically generate a time-lapse are among the features that come together to create a printer that somehow feels greater than the sum of its parts.

The *Bambu Studio* software is intuitive enough for beginners but also offers advanced functionality for power users, while the *Bambu Handy* mobile app is perfect for remotely checking the machine's webcam.

In addition to the printer and required printing parts, the X1-Carbon also includes a glue stick, lubrication for the motion system, and three 250g spools of material (PLA, PLA-CF and Support for PLA). Assembling the X1-Carbon (including the AMS) takes about 30 minutes.

The X1-Carbon uses a combination of LIDAR and analogue force sensor to measure calibration and set the Z-offset when printing. Levelling is automatic and can be done before each print to get a consistent first layer.

A filament spool bracket on the rear can hold a single spool of filament that feeds directly into the printer. Using the onboard display, filament can be loaded from either the spool holder or the top-mounted AMS. The X1-Carbon prompts you to indicate the type of material and colour when loading.

The AMS is a sleek piece of hardware, with only two visible connections (one PTFE, one wiring terminal) on the rear of the unit. It can detect filament tangles, switch materials upon runout, and use an RFID-based system to detect what material type and colour are loaded (Bambu Lab material only).

*Bambu Studio* (based on *PrusaSlicer* and *SuperSlicer*) is powerful slicer software that offers printer management, a live video feed, colour painting and many



For a larger model, you, get what you pay for.

other features. The Color Painting mode in *Bambu Studio* enables you to select areas with a variety of tools (height, fill, sphere and so on) that are used to indicate which material to use. *Bambu Studio* automatically creates a purge tower by default, as well as calculating the amount of purged material between each colour.

Running our Speed Benchy print, the X1-Carbon is unsurprisingly towards the top, with a print time of 15 minutes and 29 seconds. The Benchy that was printed on the X1-Carbon shows a few speed-related defects. The hull has a clear band where the transition to solid layers occurs, and there is some banding around the cabin. Overall, it's still an impressive print considering the sub-16 minute print time.

The X1-Carbon is not an open source 3D printer. If you are interested in the philosophical side of 3D printing, it can be aggravating to see a company 'taking' from open source ideas but not sharing developments back with the community. It's hard to find as capable a printer in this price range that offers such clearly compelling value. Not just a high-temperature material 3D printer, the X1-Carbon offers something for everyone with easy colour painting in the slicer, fast and simple setup, and advanced sensor-rich functionality. **LF**



Not the smoothest of prints but one of the fastest.

## VERDICT

**DEVELOPER:** Bambu Lab  
**WEB:** <https://bambulab.com>  
**PRICE:** £1,319

FEATURES	9/10	EASE OF USE	9/10
PERFORMANCE	9/10	VALUE	9/10

It's unusual to find a 3D printer that deserves the hype, but this is one of the best prosumer desktop FFF 3D printers around.

» **Rating 9/10**

## NTFY

# Create notifications from smart sensors

**Les Pounder** loves building gadgets with the low-cost Raspberry Pi Pico W – and some of them, like this one, even work.



**OUR EXPERT**

**Les Pounder** is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his adventures and projects at <http://bigles.com>.

### YOU NEED

- > **Pi Pico or Pico W**
- > **Android or Apple phone or tablet**
- > **Breadboard**
- > **3x M2M wires**
- > **DS18B20 temp sensor (waterproof optional)**
- > **4.7kOhm resistor (yellow - purple - red - gold)**
- > **Screw terminal (optional)**
- > **Code: <https://bit.ly/lxf314temp>**

**G**etting notifications on our smart devices is so easy – almost too easy, in that they become an annoyance. But what if you really need a notification from one of your projects? We've been using *Ntfy* for a few months now and it is super-simple to use. No fancy Python modules are required. We just need an Android or iOS app, and we're ready to go. We can send text, images and even files using the service.

In this tutorial, we're creating a simple notification system that sends hourly temperature readings directly to our Android device. The sensor, a DS18B20, has been chosen because it can be used without any software installation. The DS18B20 also comes in a waterproof variant, useful for checking water or ground temperature without damaging the sensor.

### Building the circuit

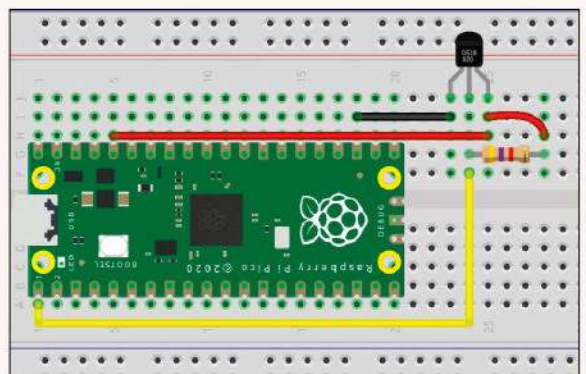
The circuit is extremely simple, but there is one gotcha to look out for. The DS18B20 has three wires: red (3V), black (GND) and yellow (one wire data). Red connects to the 3V supply on the Pico, black to any GND, and yellow goes to pin 0 (top-left). But the yellow wire also needs a 4.7kOhm resistor connecting it to 3V. This pulls the pin high and ensures that the data connection is always on. See the diagram (above-right) to see this connection illustrated. Also take a look at the high-resolution picture in the download, which shows a hack involving a screw terminal that makes the connection transferable to other devices.

### Setting up Ntfy

To use *Ntfy*, we need an Android or Apple device. Download the app from the app store and open it. We need to subscribe to a topic, so click on the + in the bottom-right. In the Subscribe To Topic dialog, set the topic name. We used `lxf-temp`, but remember that this is a public topic, so anyone can see the output. Make the topic name unique and click Subscribe. The topic page appears, and all of the messages for our topic are archived here. Go back to the homescreen of your phone, so notifications appear when we test the code.

### Prep the Pi Pico W

While holding the BOOTSEL button, connect your Raspberry Pi Pico to your computer. Now go to



The circuit is simple, but watch out for the 4.7kOhm resistor between data and 3V. We need that!

<https://bit.ly/LXF314smart> and download the version of MicroPython for your Pico or Pico W. Open your file manager, go to the downloaded file and copy it to the root of the **RPI-RP2** drive. This flashes the new firmware to the Pico.

Using your distro's package manager, install *Thonny*. For the latest Ubuntu release, we have to use a Snap:

```
$ sudo snap install thonny
```

Open *Thonny* and connect the Pico to your machine. Go to Tools > Options and select the Interpreter tab. Set the interpreter to MicroPython (Raspberry Pi Pico) and set the Port to match the location of your Pico. Click OK. *Thonny* now connects to the board and we can start writing code.

### Writing the code

The code for this project requires a few modules to be imported – luckily, they come as standard on MicroPython. The first imports are `machine` (for the GPIO), `onewire` (for the sensor), `ds18x20` (a helper module to work with the sensor) and `time` (to control the timed loop). Then we import `network` to use Wi-Fi, and `urequests` to send HTTP requests to the *Ntfy* service. We rename this as `requests` for ease of use.

```
import machine, onewire, ds18x20, time
import network
import urequests as requests
```

Next we need to create an object called `wlan` to act as a software-to-hardware connection for Wi-Fi. We



turn the connection on, then make a connection to our Wi-Fi access point. A short pause helps connection success; leaving this out means we get some connection failures. Finally, we print the connection status. If `True` is returned, then the connection is made; `False` means no connection.

```
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("WIFI AP SSID", "YOUR WIFI PASSWORD")
time.sleep(5)
print(wlan.isconnected())
```

Now we need to tell MicroPython where to find the DS18B20 sensor. It is connected to pin 0 (top-left of the Pico), so we create an object, `ds_pin`, to reference it. Then we create another object, `ds_sensor`, and using the `ds1x20` helper and `onewire` modules, we can make a connection.

```
ds_pin = machine.Pin(0)
ds_sensor = ds1x20.DS18X20(onewire.OneWire(ds_pin))
```

DS18B20 sensors have a ROM number. These are unique to each sensor and that means we can have multiple sensors on the same onewire bus. If we buy clone sensors, there is a risk that they will use identical numbers. In this case, it doesn't matter as we have just one sensor. We scan the onewire bus, and store the result in the `roms` object.

```
roms = ds_sensor.scan()
```

The `while True` loop will run the main body of code, and the first step is to sample the temperature. This is required before we can get a reading. Add a short pause to ensure that the sensor is stable before moving onwards.

```
while True:
    ds_sensor.convert_temp()
    time.sleep(1)
```

Inside the `while True` loop is a `for` loop, which reads all of the ROMs (DS18B20 sensors) on the onewire interface. We read the temperature in degrees Celsius and save the output to an object, `temp_c`, which we print to the Python shell for debug.

```
for rom in roms:
    temp_c = ds_sensor.read_temp(rom)
    print(temp_c)
```

Create an object, `temperature`, and in there store the current temperature (`temp_c`) converted to a string. Tag on to the end a space and C to format the temperature as 19.5 C, for example.

```
temperature = str(temp_c) + " C"
```

Using `requests`, we send data to `Ntfy` services using an HTTP POST request. We need to format this in order for `Ntfy` to understand it. First we need to supply the topic name – this should match what we set in the app. Next is the data that we want to send, in this case it is the `temperature` object, which contains the formatted temperature string.

```
requests.post("https://ntfy.sh/lxf-temp",
              data=temperature,
```

The next part of the request is the headers. These are extra features that we can pass to `Ntfy`. In this case, we set the title of the notification to "Hourly Temperature Check" and set it to priority 5 (high




Send notifications from your Pico W to your smartphone or tablet without all the faff!

priority). The `Tags` option enables us to use an emoji in the notification. We used the thermometer as it was in keeping with the project.

```
headers={
    "Title": "Hourly Temperature Check",
    "Priority": "5",
    "Tags": "thermometer",
}
```

Outside of the requests `for` loop but still inside the main `while True` loop, we use a `sleep` of 3,600 seconds (one hour) to pause the code between readings.

```
time.sleep(3600)
```

Save the code to the Pi Pico W as `main.py`. By doing this, MicroPython automatically runs the code when it is powered on. Power-cycle the Raspberry Pi Pico W and check your mobile device for a `Ntfy` notification. Every hour you get the latest temperature sent directly to your device. 

## » HANDY NTFY REFERENCE GUIDES

It is fair to say that we have fallen in love with `Ntfy`. Its mix of ease of use and powerful features is intoxicating and it has led us down many project paths. The basic guide on how to publish a message is at <https://docs.ntfy.sh/publish/> and there we can find examples for command line, JavaScript, Go and Python.

We are shown how to add an image to the notification, so using `Ntfy` with the Raspberry Pi camera means we can send notifications if a sensor is triggered – if the dog gets into the dining room, for example. To attach an image, we add `Attach` to the headers and then pass the path to the image. Note that you need the absolute path, be it a local file or served from the internet.

What if we want the option to action something when the notification arrives? The `Ntfy` example is a dog ringing a doorbell (go with us on this). The notification tells you that the dog wants to gain entry, but the door is locked. The `Actions` option in the header means we can trigger an event to happen. In this case, the dog is let in after calling the doorbell's API via HTTP (<https://docs.ntfy.sh/publish/#action-buttons>).

Finally, if you want to use cool emoji, `Ntfy` has a list that you can crib from. Just copy the text to the project code and the emoji is sent with your message (<https://docs.ntfy.sh/emojis/>).

» GET YOUR Pi FILLING HERE Subscribe now at <http://bit.ly/LinuxFormat>



# The Grapes of Tux

Sommelier **Jonni Bidwell** shows you how to get your pesky Windows applications running on Linux, as he uncorks the latest Wine release.

**I**f there's a Windows program stopping you switching to Linux, or stopping you doing so full time, now is the time to see if it works with *Wine*. No, we're not advocating drowning your sorrows or drinking on the job (*uh oh—Ed*). We're here to tell you about the powerful *Wine* application, which translates Windows function calls to Linux ones, thus enabling your Windows programs (be they serious tools or relaxing games) to run on our favourite operating system. That's the theory, anyway. In practice, things may need some tweaking. Or they may not work at all; *Wine* can't work miracles yet.

Be that as it may, there is a brand new release to try: *Wine 9.0*. It may be the solution to all your problems. If you've never touched *Wine* before, we'll show you everything you need to get started. And if you're a

seasoned user, we're here to show you everything that's new – including Wayland support and the improved Vulkan renderer. We've also got some tips for when things don't work, so you don't hit the bottle.

For gamers, we'll show you how to use *Proton* – the *Wine* fork from Valve Software used in its Steam platform and on its Steam Deck. Valve has gone to great lengths to ensure popular titles work with *Proton*. So, if your favourite game doesn't work with *Wine* proper, there's a good chance *Proton* can sort it.

## A fine bouquet

Readers (and writers) may be no strangers to one form or another of *Wine*, but for this outing, we're talking about the fantastic bit of software that lets an awful lot of Windows software run hassle-free on Linux. To



avoid disappointment, we'll let you know right now that this magic doesn't extend to running the newest editions of *Microsoft Office* (the one from 2010 reportedly works) or the *Adobe Creative Cloud* suite, although the commercial *CrossOver Office* from Codeweavers ([www.codeweavers.com](http://www.codeweavers.com)) does work. Profits from *CrossOver Office* directly fund open source *Wine* development.

*Wine 9.0* was released in January 2024. So, for the first time ever, the major version number rhymes with the project title. And we're here to show you what's new and how to run this hitherto hapax legomenon of a release [when *Wine 29* comes out, I'm going to come back to these parentheses and point out that your terminology is wrong – ed]. That quark of Greek being awkwardly shoehorned into place, we should also note that at the time of writing, the development release 9.3 is also available. Folks who want to live on that bleeding edge will be catered for in a moment. We're already getting ahead of ourselves.

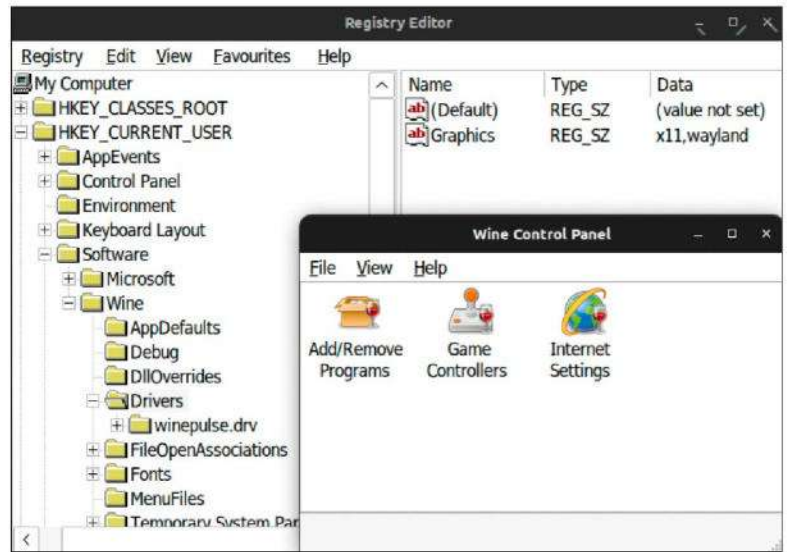
## Drinking games

For a long time, one of the most popular use cases for *Wine* was running Windows-only games. And in particular doing so through Valve's *Steam*. This was a very messy affair, and thankfully one for which there shouldn't really be any need these days (except perhaps for older 2D titles). This is thanks to *Steam* incorporating its own *Wine* implementation, dubbed *Proton*. *Proton* is the underlying technology for the *Steam Play* feature, which is what allows so many titles to work on the *Steam Deck*, Valve's portable gaming device, which runs a custom version of *Arch Linux*.

*Proton* is special in that it incorporates a magical layer called *DXVK* (<https://github.com/doitsujin/dxvk>), which allows *DirectX* (versions 9 to 11) calls to be converted to *Vulkan*. This obviates what for a long time was hobbling *Wine* – that translating *Direct3D* calls to *OpenGL* was at best clunky and at worst impossible. So, if you want to run Windows-only games, you might want to just install *Steam* instead. You can use *Steam* to run non-*Steam* titles, too. Alternatively, you might prefer to use the open source *Lutris* (<https://lutris.net>) or *Heroic Games Launcher* (<https://github.com/Heroic-Games-Launcher/HeroicGamesLauncher>), both of which enable you to run custom *Wine/Proton* versions. Or even your own manually installed *DXVK* if you really must.

Gamers will be keen to point out that AAA titles these days all use *DirectX12*. But fear not, *Wine* now has its own subsystem, *vkd3d*, for dealing with translation from *DirectX12* to *Vulkan*. This subsystem has a counterpart in *Proton*, and that fork, imaginatively titled *vkd3d-proton*, saw a new release, 2.11.1, in December 2023. This included fixes for such glitzy titles as *The Callisto Protocol*, *Halo: Infinite* and *Pioneers of Pagonia*. We'll need to give Valve some time to rebase *Proton* on *Wine 9*, but we're pretty confident that experimental versions will start appearing in *Steam* in the second quarter of 2024 – in fact at the time of publication *Proton 9.0 beta* was currently going through its release testing.

To muddy the waters slightly, *Wine* has its own native support for *Direct3D*, which is named *wined3d*. However, this is focused more on API emulation rather



than actually getting games to work. All the same, *Wine 9.0* includes improved support for *DirectX10* effects, as well as miscellaneous *DirectShow* and *DirectSound* enhancements.

*Wine* has its own Registry Editor and (rather minimal) Control Panel, styled just like in Windows.

## Wine on Ubuntu

If you're reading this from the not-too-distant future, *Ubuntu 24.04* will already have been released, and *Wine 9.x* will already be incorporated in its repos. So, you can install it directly from the *Software Centre*. But if (like us) you're still running *Ubuntu 22.04* (or a derivative of it, like *Mint 21*), you need to install *Wine* by hand. For other distros (including *Debian*), it's best to refer to the official instructions at <https://wiki.winehq.org/Category:Distributions>. Since *Ubuntu* no longer supports 32-bit installs, you need to explicitly enable the i386 architecture with *Dpkg*:

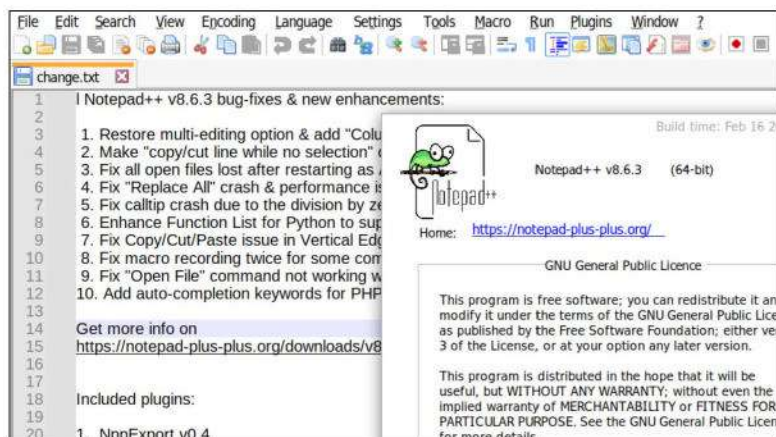
```
$ sudo dpkg --add-architecture i386
```

There are still 32-bit packages in the official repositories (for example, the libraries required for *Steam* and others), it's just *Apt* will refuse to install

## » IN VINO VERITAS

*Wine* isn't an emulator. That's what *WINE* used to stand for back when it was an acronym. So, if for some reason you have some Windows on ARM binaries you want to run on your x86 system, you can't do it in a straightforward way. However, *Wine* can be built for ARM64 devices and you can even find official builds in the Raspberry Pi OS repos. These are rather old, though, and you need to use the packages from the Debian Unstable (Sid) repos instead (see <https://packages.debian.org/sid/arm64/wine64/download>). By adding an emulation library such as *FEX* (<https://fex-emu.com>), it's even possible to run x86 binaries on ARM devices.

*Wine* is truly cross-platform, in that it is available for Mac OS and ChromeOS (yes, we know that's just Linux with a Google-veneer). Mac users are spoiled with their M1, M2 and M3 CPUs, and Apple's proprietary Rosetta code translator (that allows x86 code to run speedily on said silicon). But a faction of those users still take umbrage with Apple's decision to banish 32-bit applications since the Catalina release in 2019. Thanks to *Wine's* new *WoW64* mode (see later), those users can finally have their way without resorting to virtualisation through *Parallels*.



Some Windows users swear by Notepad Plus. Now you can swear by (or at) it in Linux.

them unless the archive is activated. *Wine* and *Steam* both allow 32-bit programs to run, and as such rely on 32-bit libraries. But they both allow you to run 64-bit programs (in entirely different ways), making the whole situation far too complicated to wade into right now. Instead, let's add a directory to house external keyrings and add the WineHQ archive key to it:

```
$ sudo mkdir -pm 755 /etc/apt/keyrings
$ sudo wget -O /etc/apt/keyrings/winehq-archive.key https://dl.winehq.org/wine-builds/winehq.key
```

## HIT THE RIGHT NOTE

“Explore the application and puzzle at why people prefer Notepad++ to Vim if you're not familiar with it.”

Now we can add the WineHQ repo to Apt:

```
$ sudo wget -NP /etc/apt/sources.list.d/ https://dl.winehq.org/wine-builds/ubuntu/dists/jammy/winehq-jammy.sources
```

If you're using the previous Ubuntu LTS release, 22.04 (Focal Fossa), you need to substitute `focal` in the two occurrences of `jammy` above. Or, if you're using Ubuntu 23.10 (Mantic Minotaur), substitute `mantic` instead. You should also be aware that 23.10 will go

end of life (EoL) in July 2024. Finally, we can update our package lists and install *Wine* from its own repo:

```
$ sudo apt update
$ sudo apt install --install-recommends winehq-stable
```

It's important to note that installing *Wine* this way can be a little hit or miss. Which is a polite way of saying that if you blindly confirm the last command, you can royally trash your system. In particular, check that it doesn't want to remove important-sounding packages (such as `ubuntu-desktop`). If it does, cancel the install and see if you can find up-to-date guidance on the official forums at <https://forum.winehq.org/viewforum.php?f=8>. On our nearly clean install of Ubuntu 22.04, the install worked fine, but it did pull in some 300MB of dependencies, so we were careful to study the extensive output before confirming.

The Stable release should be fine for most people, but if you really must have all the latest features, you might want to try the Development release instead. In that case, install the package `wine-devel` instead. It's possible to have both versions installed side by side, but the manual advises using different prefixes (directories where *Wine*'s per-user configurations are stored, see later) for each. Besides the Development release, there's the absolute bleeding-edge Staging version; this includes some bug fixes not present in the dev version, but probably some new bugs, too. It's also where new features are first staged, hence the name. If you think this is for you, grab `wine-staging`.

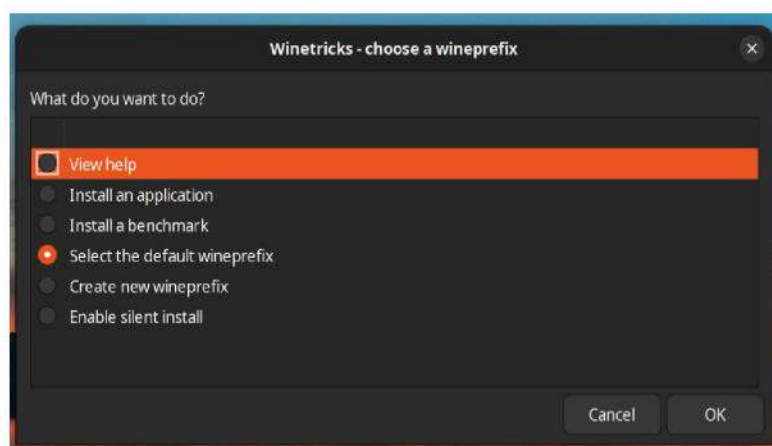
## Tasting Notes

We'll start simple and show you how to get the popular *Notepad++* text editor running. Because of its popularity, Canonical already offers a Snap of this package (you'll find it in the Snap Store), but behind the scenes, it's still using *Wine*. You might not be using Ubuntu, or you might just not like Snaps. In which case, do read on. Grab the 64-bit (unless you really want the 32-bit one) installer from <https://notepad-plus-plus.org/downloads/> then run:

```
$ cd ~/Downloads
$ wine npp.8.6.3.Installer.x64.exe
```

Update the version number to match what you downloaded. You'll see some odd-looking (but mostly harmless) output in the terminal. If this is your first time running *Wine*, then the Mono (an open source re-imagining of Microsoft's .NET framework) installer will spring into life. Then you'll be able to click through the *Notepad++* install as if you were in Windows. *Wine* maintains its own virtual `C:\` drive, so all the standard Windows directories are there. That drive is stored inside the default *Wine* prefix (the hidden `.wine/` directory in your home folder – press `Ctrl+H` in Files if you can't see it) at `/home/user/.wine/drive_c/`. The installer gives you the option to start *Notepad++* right away, so do that and you should see that everything looks roughly as it would within Windows. Yes, the fonts might be a bit fuzzy, but generally speaking, it's all there. Make a text file and save it if you want, or explore the application and puzzle at why people prefer it to *Vim* if you're not familiar with it.

Quit *Notepad++*, open a terminal and run `Winecfg`. This is *Wine* configuration tool, where the current prefix can be tweaked. Again, you may see some



You can use the Winetricks script from the command line, or through this fetching wizard.





peculiar warnings as it starts, and again these are nothing to worry about. The Applications tab lets you choose which version of Windows to 'emulate', and this can be configured on a per-application basis. We won't get into substituting libraries here, but it is possible to specify whether to use Wine's built-in version of a particular DLL (dynamic link library) or a native file from Windows. We will say that it's a terrible idea to use Windows files for low-level stuff (such as `gdi32.dll`, `ntdll.dll`, `kernel32.dll`). Apart from one-off library substitutions, it's much better to use the Winetricks script (see box) to manage libraries.

The Drives tab shows you virtual drives and enables you to set up additional ones (for example, for an ISO-mounted DVD drive). Besides the **C:** drive, you'll see a **Z:** drive, which is your root filesystem. If you investigate this further (seeing where *Notepad++* can save a file, for instance), you'll see that Wine has access to the same bits of the filesystem as you do. So, if you were to run some Windows malware, it could very well trash your **home** directory (or other places where you have write access). The other tabs in *Winecfg* are best explained by the documentation.

Next, let's check out the big news with this release, namely Wine's improved Wayland support. If you're not running Wayland, this won't be of much interest, but Ubuntu (where Wayland has been the default for some time, even on Nvidia cards) users in particular would be well advised to check this out. Whatever your distro, first check that you're indeed running Wayland with:

```
$ echo $XDG_SESSION_TYPE
```

Hopefully, this says **wayland**. If not, log out and see if your login screen gives you a choice of Wayland or X.org. To enable the Wayland support, we need to edit a registry key. Windows aficionados might want to use Wine's own Registry Editor (see screenshot – p.45 – or run `wine regedit` to see it for yourself), but it's probably easier to just type:

```
$ wine reg.exe add HKCU\\Software\\Wine\\Drivers /v Graphics /d x11,wayland
```

We also need to unset the **\$DISPLAY** environment variable, which is an X11-thang:

```
$ unset DISPLAY
```

There isn't anything else user-centric to configure here, but with this enabled, Wayland users should see better HiDPI support, have more luck when using multiple monitors and enjoy better desktop integration. In particular, titles exhibiting strange behaviour when using mouse-look should behave much better. Wine's (still experimental) Vulkan driver now supports Vulkan

1.3.272, so Windows titles using new Vulkan extensions there will work and look nicer. See the Winetricks box (right) for instructions to enable the Vulkan renderer.

We avoided discussion of 32 vs 64-bit executables earlier, but now we're settled in, we should note that Wine now handles 64-bit Windows binaries in a much more robust way. This is part of a years-long effort to revamp the WoW64 (Wine on Windows 64) componentry. This allows 32-bit programs to run without 32-bit libraries on the host system. And the so-called new WoW64 mode also allows such apps to run on a purely 64-bit system. This mode isn't yet enabled by default (which is why our instructions add the i386 architecture), and to use it you have to build from source. The default old WoW64 mode runs 32-bit apps inside a 32-bit process. The Wine devs state that while the new mode should be fine, old mode is better for 16-bit stuff (maybe you prefer the Windows 3.1 version of *Paint*) and OpenGL programs. **LXF**

Steam Play is enabled by default on Linux for supported titles, but flip this switch if you want to try it on others.

## » WINETRICKS AND PREFIXES

If you do enough experimentation, you'll soon find a Windows program that doesn't seem to work with Wine. In this instance, your first port of call should be the Wine Application Database (AppDB, see <https://appdb.winehq.org>). It's a user-curated list of applications and how well they work with Wine, with a ratings system and remedial guidance for programs that require a little extra persuasion. Generally, this persuasion involves using Windows native libraries instead of those provided by Wine. Common offenders are the .NET and Visual C runtimes, various codecs and the quartz.dll library, which some games use for video.

Rather than mess around substituting one library at a time, we can use the Winetricks helper script to automate these fixes. Entries in the Wine AppDB often include instructions for Winetricks. An old iteration is included in the Ubuntu repos, so you're better off following the manual installation instructions at <https://github.com/Winetricks/winetricks>.

Wine's experimental Vulkan renderer needs to be activated by adding a registry key. But we can save several clicks by getting Winetricks to do this for us:

```
$ winetricks renderer=vulkan
```

By default (unless you compiled Wine yourself), Wine sets up a 64-bit prefix at `~/wine`. It is absolutely possible to run 32-bit applications from here. But if, for whatever reason, you want to set up a 32-bit prefix at `~/wine32`, you can do so (taking care not to confuse your wins and wines):

```
$ WINEARCH=win32 WINEPREFIX=~/wine32 winecfg
```

# Audit your system for vulnerabilities

Invulnerable Shashank Sharma has the perfect tool to keep systems safe.



**OUR EXPERT**

**Shashank Sharma** is a trial lawyer in New Delhi, India. He's been writing about open source software for 20 years and lawyering for 10.

**S**ecurity audits are like hobbies. You always hear good things from people about them, and your life is vastly improved if you put one to good use. But if you're anything like us, you've convinced yourself that performing a security audit on your system is a time-consuming exercise, and one that only seasoned veterans can carry out. Thankfully, with the right tool at your disposal, you can test your system for vulnerabilities in almost no time at all.

Released under the GPLv3 licence, *Lynis* is a security audit tool designed to test your system across a vast array of categories, such as Boot and Services, Kernel, Memory and Processes, Shells, Filesystems, USB Devices, Storage, NFS, Printers and Spools, Networking, SSH Support, Databases, Users, Groups and Authentication, and more. While there are various tools that can be deployed to test the vulnerabilities of any one of these categories, *Lynis* stands out because of its ability to test your system across so many areas.

Based on the tests it runs, *Lynis* offers suggestions and recommendations to help you address any

vulnerabilities that could potentially be exploited to attack your system.

### Security audit for the masses

Although the project is aimed primarily at system administrators, security auditors/officers, penetration testers and security professionals, its ease of use makes it possible for just about all manner of Linux users to put it to into service.

You'll find *Lynis* in the software repositories of most popular Linux distros. Run the `sudo apt install lynis` command if you're running a Debian, Ubuntu or derivative distro. Run the `sudo dnf install lynis` command instead if you're on an RPM-based distro.

Depending on your distro of choice, the version of *Lynis* on offer might be a little dated. Considering the nature of the project, it makes sense to always run the latest version. To grab the most up-to-date iteration from the project's GitHub page, run the `git clone https://github.com/CISOfy/lynis /` command. This downloads the latest version into the `lynis/` directory.

The tool doesn't require any installation and you can test your system straight away by navigating into the `lynis/` directory and running the `./lynis audit system /` command. But we'll get around to that in just a bit. For now, let's get acquainted with the tool and its functions.

From the downloaded `lynis/` directory, run the `./lynis show commands` command for a list of supported commands. You can also run the `./lynis show help` and the `./lynis --man` commands to get acquainted with the tool (see table, opposite page).

When you run the `./lynis audit system` command, the tool audits your system across its entire range of tests. While running the audit, *Lynis* generates a lot of output. Once the audit is complete, towards the bottom you'll find a list of suggestions and details of the scan. For instance, on our test machine running Ubuntu 20.04 LTS, *Lynis* ran 255 tests, with a hardening index

```
1: linuxlala@playground: ~/Downloads/projects/lynis
Lynis security scan details:

Hardening index : 63 [##### ]
Tests performed : 255
Plugins enabled : 2

Components:
- Firewall           [V]
- Malware scanner    [X]

Scan mode:
Normal [ ] Forensics [ ] Integration [ ] Pentest [V] (running non-privileged)

Lynis modules:
- Compliance status [?]
- Security audit     [V]
- Vulnerability scan [V]

Files:
- Test and debug information : /home/linuxlala/lynis.log
- Report data                : /home/linuxlala/lynis-report.dat
```

In addition to providing tips for general system hardening, *Lynis* also informs you of vulnerable software packages and possible configuration issues.



Command	Description
<code>./lynis --man</code>	Print the man page
<code>./lynis show help</code>	Print the help screen
<code>./lynis show</code>	Print all possible arguments
<code>./lynis show commands</code>	Print all supported commands
<code>./lynis show help &lt;command&gt;</code>	Print help for specified command
<code>./lynis show settings</code>	Print the default settings
<code>./lynis audit system</code>	Run audit on the current system
<code>./lynis audit remote &lt;host&gt;</code>	Run audit on specified remote system

of 65 out of 100. Thankfully, there were no warnings, but the tool did give us 40 suggestions for to improve the security of our system.

## Understanding the tests

Due to space constraints, we can only list a few of the warnings published by *Lynis* after the security audit:

- \* If not required, consider explicit disabling of core dump in `/etc/security/limits.conf` file [KRNL-5820]  
<https://cisofy.com/lynis/controls/KRNL-5820/>
- \* Install a PAM module for password strength testing like `pam_cracklib` or `pam_passwdqc` [AUTH-9262]  
<https://cisofy.com/lynis/controls/AUTH-9262/>
- \* Configure maximum password age in `/etc/login.defs` [AUTH-9286]  
<https://cisofy.com/lynis/controls/AUTH-9286/>
- \* To decrease the impact of a full `/home` file system, place `/home` on a separate partition [FILE-6310]  
<https://cisofy.com/lynis/controls/FILE-6310/>
- \* Purge old/removed packages (295 found) with `aptitude purge` or `dpkg --purge` command. This will cleanup old configuration files, cron jobs and startup scripts. [PKGS-7346]  
<https://cisofy.com/lynis/controls/PKGS-7346/>
- \* Double check the permissions of home directories as some might be not strict enough. [HOME-9304]  
<https://cisofy.com/lynis/controls/HOME-9304/>
- \* Harden the system by installing at least one malware scanner, to perform periodic file system scans [HRDN-7230]  
- Solution : Install a tool like `rkhunter`, `chkrootkit`, `OSSEC`, `Wazuh`  
<https://cisofy.com/lynis/controls/HRDN-7230/>

Refer to the Improving Security box (see right) on how to use these suggestions to harden your system and protect yourself.

```

1/1 + + + ...:linuxlab@playground: ~/Downloads/projects/lynis
1:linuxlab@playground: ~/Downloads/projects/lynis
- Malware software components [ NOT FOUND ]

[+] File Permissions
- Starting file permissions check
File: /boot/grub/grub.cfg [ OK ]
File: /etc/crontab [ SUGGESTION ]
File: /etc/group [ OK ]
File: /etc/group [ OK ]
File: /etc/hosts.allow [ OK ]
File: /etc/hosts.deny [ OK ]
File: /etc/issue [ OK ]
File: /etc/issue.net [ OK ]
File: /etc/passwd [ OK ]
File: /etc/passwd [ OK ]
File: /etc/ssh/sshd_config [ SUGGESTION ]
Directory: /etc/cron.d [ SUGGESTION ]
Directory: /etc/cron.daily [ SUGGESTION ]
Directory: /etc/cron.hourly [ SUGGESTION ]
Directory: /etc/cron.weekly [ SUGGESTION ]
Directory: /etc/cron.monthly [ SUGGESTION ]

[+] Home directories

```

Although useful, implementing all the suggestions would be overkill for most home users.

The output generated by *Lynis* is colour-coded for quick assimilation of important information. Green is used to confirm the presence of a scanned module or component. This is for information purposes only and doesn't require further any action. Orange implies suggestions that the tool wishes you to look up and address. It could mean that the module or component is disabled. Red is reserved for components that require immediate attention.

Some tests require root privileges, but you'll face an error when you run the `sudo ./lynis audit system` command. Instead, you have to first establish ownership of the file with the `chown -R 0:0` command and then run *Lynis* with `sudo`:

```

$ cd Downloads/projects
$ sudo chown -R 0:0 lynis
$ cd lynis
$ sudo ./lynis audit system

```

As well as running an audit of the local system, you can also audit a *Docker* file. Considering its ease of use and the infinitely useful information generated, *Lynis* is now on our list of highly recommended tools. **LXF**

## QUICK TIP

To ensure security of your installation, you must always use anti-malware and antivirus tools, and it's good practice not to use the same password across different accounts. For most home users, these precautions are enough to safeguard your system. After all, disabling USB support at the behest of an audit tool can land you in the soup with your family.

## » IMPROVING SECURITY

For each suggestion, *Lynis* provides a useful description along with a TEST-ID, such as FILE-6310. You can use these identifiers to read more about the test with `./lynis show details <test-ID>`. Alternatively, you can click the URL provided to read more on *Lynis*'s website.

While some of the suggestions are easy enough to address, even for newcomers, such as the one recommending purging old packages or installing a malware scanner, others might require a bit more effort on your part to understand and implement.

Throughout the output generated, a green [V] denotes a tick, and a red [X] suggests an absence. On our machine, we have configured a firewall but don't have a malware scanner, so *Lynis* shows us this:

Components:

- Firewall [V]
- Malware scanner [X]

You don't have to scroll backwards through your terminal screen to access the results of the tests run by *Lynis* or the output it generated. You'll find the same in the `~/lynis-report.dat` data file. The data file, however, isn't colour-coded, unlike the real-time output generated by *Lynis*.

» **ENHANCE YOUR TERMINAL-FU** Subscribe now at <http://bit.ly/LinuxFormat>

## KAVITA

Credit: www.kavitareader.com

# Build a streaming ebook library

Bionic bookworm **Nick Peers** looks at a free tool that enables you to collate, read and share your digital publications over the internet.



**OUR  
EXPERT**

**Nick Peers** has more paper and digital publications than he cares to admit. Maybe *Kavita* can help the digital library look less chaotic than the piles of paper and magazines filling his office.

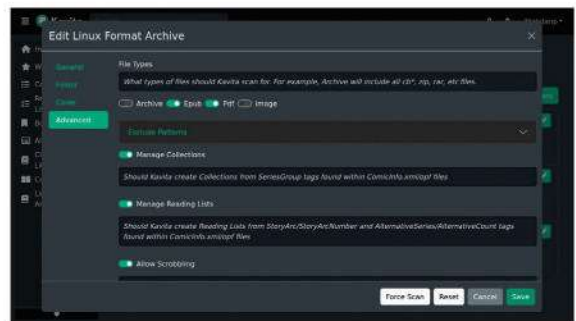
**K**avita does for your digital books and comics what *Audiobookshelf* does for your digital audiobooks – it provides easy access to all your ebooks via a web browser or app, at home and further afield. Although geared towards comic book users and manga fanatics, there's enough in *Kavita* to suit those who simply want a user-friendly home to manage, read and share their digital publications.

## Put up a bookshelf

If you want to install *Kavita* natively on your system, head to <https://wiki.kavitareader.com/en/install/linux-install> for instructions. It involves saving the correct build of the latest release at <https://github.com/Kareadita/Kavita/releases> to your hard drive, then extracting the .tar.gz archive to a writable directory of your choice, making it executable ( `chmod +x ./Kavita` ), and running the app directly ( `./Kavita` ). You'll also find instructions for installing *Kavita* as a *Systemd* service.

If you're adding *Kavita* to your server, you may prefer to run it containerised – you'll find instructions for *Docker* at the *Kavita* wiki, where there are multiple options, including a **LinuxServer.io** version for those who want to run *Kavita* under their own user account.

We've also successfully tested the official container with *Podman* as a rootless container with the following:



Each library can be configured to contain a broad range of file types, or you can restrict it to just the common formats (PDF and EPUB).

```
podman run --name kavita \
--usersns keep-id:uid=1000,gid=1000 \
-p 5000:5000 \
-v ~/ebooks:/ebooks \
-v ~/container-data/kavita:/kavita/config \
--restart unless-stopped \
-e TZ=Europe/London \
-d docker.io/jvmilazz0/kavita:latest
```

You obviously need to adapt the two `-v` lines to point to your ebooks folder on the one hand, and the location where you plan to store *Kavita*'s configuration files on the other. We've also assumed you're running *Podman*

## QUICK TIP

Looking for content with which to populate your library? You can find free books from several legal sources, including Project Gutenberg ([www.gutenberg.org](http://www.gutenberg.org)) and the Internet Archive (<https://archive.org>).

## » ORGANISE YOUR FILES

The secret to a tidy library is file organisation. *Kavita* relies on you to properly file everything in a logical manner so it can concentrate on scanning through your content. Start by organising the material into separate libraries if that's applicable – each library needs its own folder inside a parent directory.

From here, how you organise your files depends on the file type. EPUB files, for instance, just can be thrown in haphazardly

(or organised by folder and subfolder such as author or genre) – *Kavita* still treats each book as its own unique series based on its metadata.

PDF files, however, are treated differently. Because there's no metadata to speak of, *Kavita* relies on the folder structure to organise them. This organisation extends to placing each set of files in a grouped folder that is treated as a series. The folder's name then becomes the series

title, while the filename is also the book's title within *Kavita* itself.

Don't worry too much about trying to apply organisation. As you'll see from the main copy, *Kavita* provides many ways to view your collection, with smart filters the most flexible. You can also edit your library's metadata, as the step-by-step guide reveals (see page 52), ensuring even PDF files can be populated with key information like author and genre to help you organise them.



through your own user account, which has a UID and GID of 1,000.

By default, you need to manually restart *Kavita* each time you boot your PC. To avoid this, issue the following lines to set up a user-generated service:

```
$ podman generate systemd --new --name kavita -f
$ mv -v container-kavita.service ~/.config/systemd/user/
$ systemctl --user daemon-reload
$ systemctl --user enable container-kavita.service
```

## First read through

Access to *Kavita* is through your web browser – either **localhost:5000** from your PC or **192.168.x.y:5000** from other devices on your network (replace **192.168.x.y** with your PC's IP address). It's also simple to extend access beyond your network, particularly if you're running *Nginx Proxy Manager* (see **LXF306**) – just set up a proxy pointing to **http://192.168.x.y:5000**.

When you log on through your browser, you're invited to set up the administrator's account. Supply a suitable username and password (plus your email if you want to be able to reset your password at any point – this requires you to set up email after logging in) and then click to register. You're then prompted to log in.

You're shown the main screen – the annotation (right) reveals how it looks once you've got a library up and running, but for now you'll want to start adding content. Click the Server Settings link to be taken to the Admin Dashboard's Libraries tab (you can also access the dashboard via the settings button next to your username in the top-right).

Click Add Library. Start by giving it a suitable name and type (the default Manga betrays *Kavita*'s focus, but there's a Book option, too). Click Next to select the library's folder (see the box opposite for how libraries should be organised). You're next given the option of uploading a 32x32 library image (click Next to skip).

The Advanced section is where you determine what file types to search for (EPUB and PDF are pre-selected by default). In most cases, all other options can be left as they are, so click Save and you'll see the library appear with its status displayed, as well as options for refreshing (performing a new scan), deleting and editing. The latter option brings up the same four categories as before (General, Folder, Cover and Advanced), which gives you the option of adding parent folders to your library, plus tweaking settings.

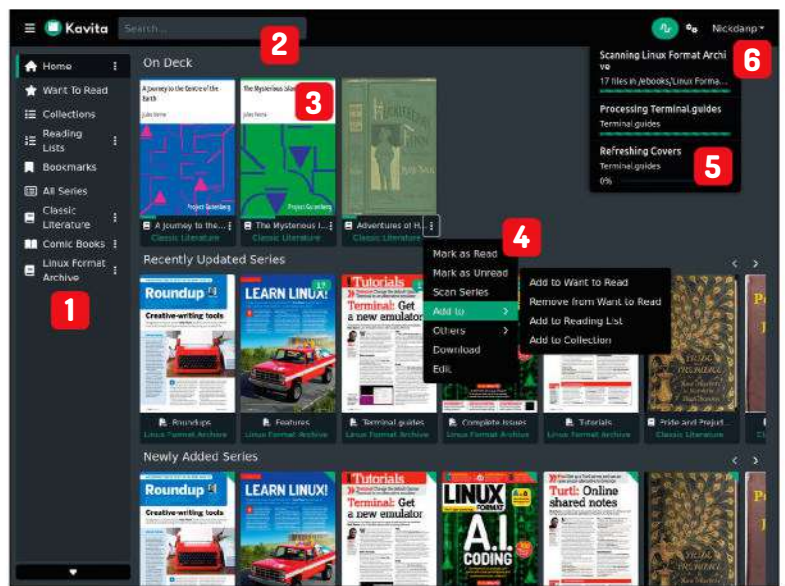
## Browse your library

You're ready to view your library. Use the navigation tools on the left to find what you're looking for. Start by



Use smart filters to build highly customised views of your libraries – these are accessible via your personal user menu.

## NAVIGATE YOUR KAVITA LIBRARIES



### 1 Navigation pane

Quickly switch between *Kavita*'s main views here – including top-level access to your libraries.

### 2 Smart search

The keyword search tool displays results as you type, split into categories such as libraries and files.

### 3 On Deck

*Kavita* tracks the progress of your books, enabling you to quickly jump back in.

### 4 Quick edit tools

Click the vertical ellipsis (⋮) button next to a title or series to access extra options.

### 5 Activity meter

When the button is green, a scan is taking place. Click the button to see what's going on.

### 6 User menu

Click your username to access user-specific settings and any smart filters that you've saved.

clicking Home to reveal a card-like view displaying recently updated and newly added series – right now, they should contain the same content from your new library. Later, when you start reading through your content you'll see an extra category – On Deck – which works in the same way as your media server does to put the content that you're reading front and centre.

On the left you'll see navigation options, including options to browse each library you add. Click a library to view all its titles – dubbed Series – in the main window. Click an individual title from here and you're taken to its homepage where a Read or Continue button lets you read it from the comfort of your browser courtesy of *Kavita*'s own document viewers.

Next to this are further buttons. The first is a star – click this to add the title to your Want To Read list. There's also a button for marking the title read or unread, followed by a metadata editing tool – see the step-by-step guide (over page) for details on how this can provide information to help navigate your libraries.

Next is the ... button, which reveals a pop-up menu. Many of these options are duplicated elsewhere, but the menu is also accessible from other views (look for the ⋮ button next to a series or title), so it's worth familiarising yourself with its various options as you can quickly add titles to other lists with fewer clicks.

Beneath these buttons you'll find information about the file (including format, length and estimated reading time). If there's any metadata tagged in the file – only applicable to EPUB files – then it may be displayed here, too, giving you access to the author (or writer as

## QUICK TIP

If you're constantly adding new content to your folder, spare yourself continually prodding the library update button by navigating to the Admin Dashboard's General tab, then scrolling down to enable the folder watching setting.



## QUICK TIP

Everyone needs a backup, and *Kavita* is no exception. It automatically backs up your database on a daily basis (you can change this via the Tasks tab on the Admin Dashboard). Visit <https://wiki.kavitareader.com/en/install> for details on where the backups get stored.

*Kavita* terms it), tags and other information. These can be used as quick search filters – click the Writer tag, for instance, and you're shown a list of all titles in your collection with that author tagged.

## A good bookmark

If there are too many titles to view, look for the filter button in the top-right corner of the main window. Click this to discover how *Kavita* displays the content of your library – via smart filters. The shortcuts on the left of each library basically create a view that displays all titles, filtered to show only those in the currently selected library.

From here you can create all kinds of different views using a combination of smart filters. You'll see filters can be based on series name, metadata (including writer and tags), file path and more. These can be combined and configured to focus on specific matches or broadened to span multiple libraries, authors or whatever you're looking for.

Filters can be given a suitably descriptive name, enabling you to save them for easy retrieval later via

the Smart Filters entry under your own user menu in the top-right corner of the *Kavita* user interface.

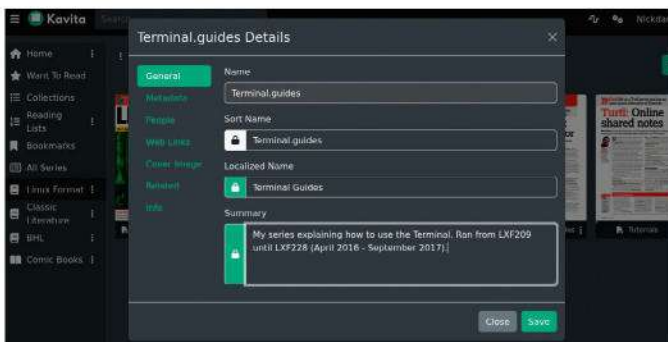
Other navigation options listed on the left include Collections, Reading Lists and Bookmarks. Collections enable you to group your choice of individual titles (or series) together. Click the vertical ellipsis (⋮) button next to a title and choose Add To > Add To Collection, then create a new collection or select an existing entry.

Reading Lists work in a slightly different way – again set them up from the Add To menu. Here, items (anything from a single chapter to an entire series) can be grouped together in a list. Once created, click Reading Lists on the left followed by the list to see your titles arranged in the order you added them. You can rearrange them via drag-and-drop, plus remove or read individual titles. Click Continue to start reading (and later pick up from where you left off). When you finish one title, the next one automatically follows on.

## Reading the words

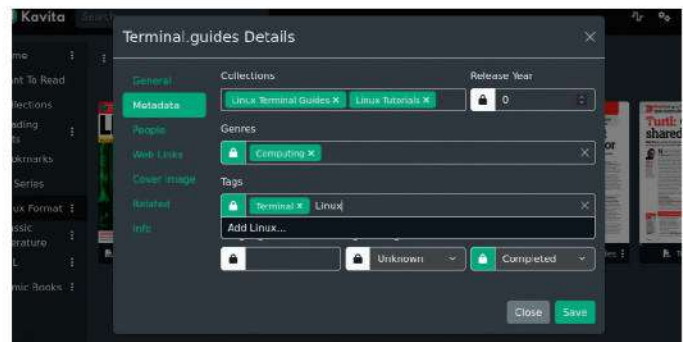
Reading a book is as simple as clicking its cover followed by the Read button. *Kavita* provides you with

## EDIT SERIES DETAILS AND METADATA



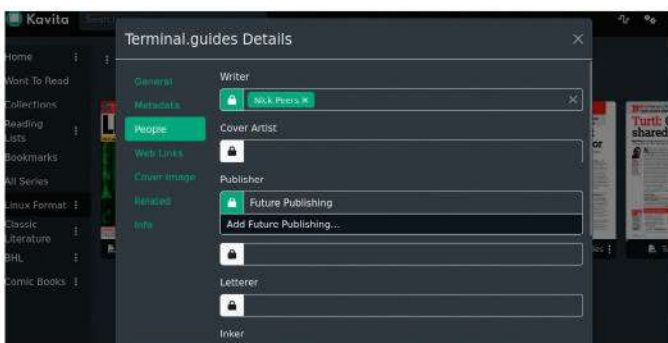
### 1 Change name and summary

Metadata can be applied to a series, but not to individual files (or entries in a series). Browse to the library containing your series, click the vertical ellipsis (⋮) button and choose Edit to bring up its Details page. Use the General tab to view its (non-editable) name plus add an explanatory summary to display on the series's homepage.



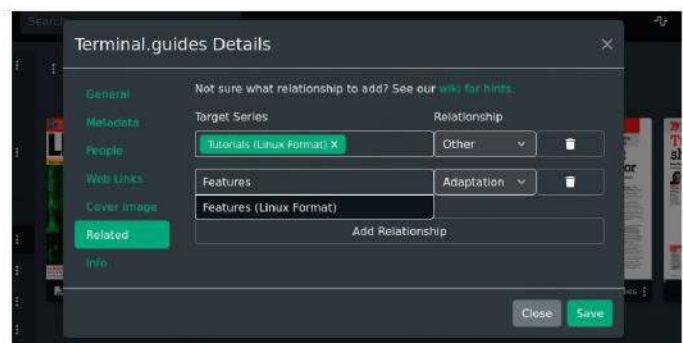
### 2 Edit metadata

The Metadata tab enables you to add the series to an existing collection (or create a new one). It's also where you can add a release/publication year, assign genres and generic tags (your choice of keywords). You can also supply a language, age rating and publication status for periodicals such as magazines.



### 3 Assign credits

The People tab is where you can build a database of contributors to the publication or book. At its basic level, you can simply supply the writer (or author), but there are many other fields, too – aimed primarily at comic books, but some can be translated to other mediums, such as Publisher, Editor and Character (for fiction series).



### 4 Add other details

The Web Links page enables you to link to any websites you wish – such as the author, publisher or magazine's own homepage, such as [Linuxformat.com](http://Linuxformat.com). Use the Cover Image to pick a different image – either from the other documents in the series, or by uploading your own. Finally, Related enables you to link the series to others.



different in-browser readers for different formats, so PDF files display differently from EPUB. In both cases, the book is displayed with its pages top-down (so scroll down to read through it). At the top you'll find a series of controls – the PDF viewer has more buttons to choose from (roll your mouse over one to reveal what it does). Use the hamburger (≡) icon in the top-right to change viewing options, such as displaying pages in spreads and scrolling horizontally between pages rather than vertically. The button on the left reveals a navigation page with page thumbnails you can scroll through to jump more quickly to where you need to go.

The EPUB reader looks barer at first glance but click the menu icon to reveal a slide-out menu, offering both comprehensive Settings and a Table of Contents view. If you double-click on a page to reveal the option required to create a bookmark, you can access those from the Table of Contents section.

In both cases, when you close the reader, *Kavita* records where you are in the book – you'll see the Read button now says Continue, and the book also appears in the On Deck section on your homepage.

For those using *Kavita* to manage comic book libraries, *Kavita's* Comic Manga Reader (which works with CBZ files you can obtain from sites like <https://archive.org/details/comics>) is the only one to work with the Bookmarks shortcut you can access directly from *Kavita's* main navigation menu. When browsing comics, press the Space bar to bring up the controls – the bookmark button is in the top-right corner.

Sadly, there's no official app for browsing your content from your desktop, phone or tablet – you can, of course, use your mobile browser (save the website as a shortcut to your home screen) or visit <https://wiki.kavita-reader.com/en/faq/external-readers> for a guide to using selected apps. One of the most versatile is *KOReader* (<https://koreader.rocks>), which you can install on Linux, Android and Kindle Legacy models.

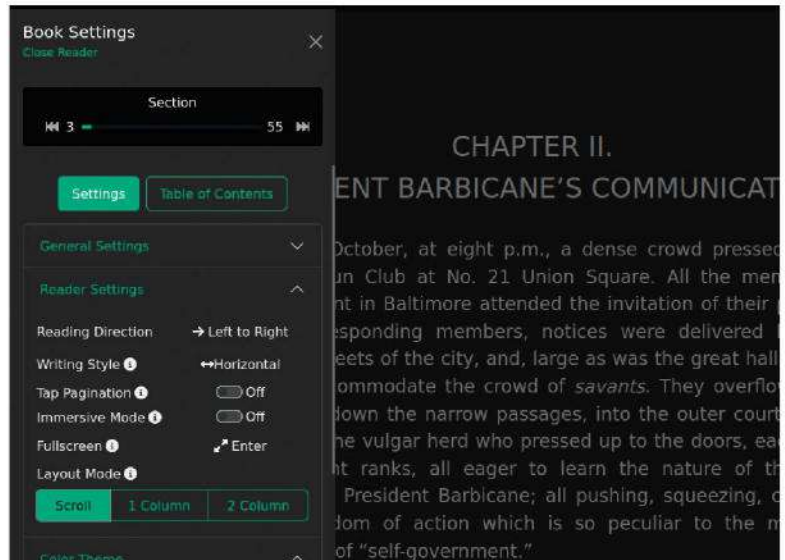
Finally, Bookmarks enable you to save individual pages for reading or downloading. This only works with *Kavita's* Comic Manga Reader; the EPUB Book Reader can save bookmarks, but only internally, while the PDF Reader offers no bookmarking tools at all.

## Server tweaks

There are two areas where you can tweak settings in *Kavita*. Your personal user preferences are accessible via the Settings option under your username at the top-right. Selecting this reveals your User Dashboard, with settings spread out over six tabs. The default Preferences tab enables you to tweak the look and behaviour of the main web interface as well as the built-in image (comic book) and book (EPUB) readers.

Visit the Account tab to change your password and email – you'll see a greyed-out Age Restriction option, which only applies to other users of your server (see *the box, right*). The 3rd Party Clients provides the API Key and OPDS URL you need to plug your *Kavita* server into third-party readers such as *KOReader*.

The Theme tab enables you to switch the interface skin but requires you to click the Theme GitHub link to find more themes to download (with instructions). Use



the Devices tab to configure ereaders – you'll see entries for Kindle, Kobo and PocketBook along with a Custom entry. Once configured, you can push books to your ereader via email.

The final tab enables you to review your activity via stats – info such as total pages read, time spent reading and average time per week mingle with the ability to analyse your reading over set periods.

The other settings button – found sandwiched between the activity meter and your username – takes you to the Admin Dashboard. Most of the options here are self-explanatory, but take the time to visit the Statistics tab for information on your library's content as well as how others are using it. **US**

Kavita's built-in readers work with PDF, EPUB and CBZ files. The EPUB reader gives you full control over your book's layout.

## » SHARE YOUR LIBRARIES

One of *Kavita's* strengths is that it enables you to share your libraries with others. If you want to share with people outside your home, you need to configure *Kavita* to work through your remote proxy.

To set this, open the Admin Dashboard via the main settings button (next to your username in the top-right corner of the *Kavita* user interface) and switch to the Users tab. Click the Invite button to set up a new user, but note the warning about email – if you've not already done so, close the invite pop-up, then switch to the Email tab to configure *Kavita* to send emails.

After inputting their email address, you can assign roles to that person – while we recommend limiting Admin to yourself or maybe one other trusted user, other roles include allowing downloads, changing passwords, setting bookmarks and setting read-only so they're unable to make changes to metadata or other settings.

Next to this is a list of all currently available libraries – you can automatically grant them access to all libraries or restrict them to specific content. Once done, scroll down to reveal an Age Rating Restriction section, perfect for restricting library access to the younger members of your household.

Once done, click Invite and an email is sent with instructions for completing the registration. There's also a Setup User button next to the user's entry under Users, which can be used to set their account up immediately.

» **KEEP READING US EACH MONTH** Subscribe now at <http://bit.ly/LinuxFormat>

## SYSTEMD

Credit: <https://systemd.io>

# After the kernel, it's startup time, init!

**Mats Tage Axelsson** explains how Linux takes you from the kernel to a running system, both today and back in the mists of time.



**OUR  
EXPERT**

**Mats Tage Axelsson** can be counted on to find new ways to simplify your Linux usage in complex ways. Anything to seem as cool as he thinks he is.

**H**ow do you start a computer? “Press the power button!” is not a valid response. Most of you know that the firmware loads the kernel, which in turn hands it over to the system. Maybe you know that it starts process one, aka PID=1, and that this process is the key not just to booting fast and securely, but also to keep all things running.

As expected with Linux, you have many options to handle your own boot. Even the distinction that PID=1 also handles services and other things is not always a given. It started with shell scripts, which users and administrators enhanced with SysV, and via other ideas we mostly now use *Systemd*.

That is almost a universal Linux truth, but nothing is ever certain in our world, so new ideas have come out. We'll cover here what changes there have been, where it got us and a hint about things to come.

## To start: history

In the early days of Unix, the system would start with *init* as process number one. The process number one is still the way your computer starts. Originally, the *init* ran shell scripts to run everything. This worked well for the small systems of the time, though even then it had limitations. One major limitation is that they run one after another. If one script fails, the whole boot fails. And when one script starts slowly, the whole boot

```
cgroup-lite.conf  ubuntu-fan.conf  whoopsie.conf
matsstage@matsstage:~$ cat /etc/init.d/
acpid              lm-sensors
alsa-utils         mpd
anacron            nginx
apparmor           openvpn
appport            pcsd
atd                plymouth
atop               plymouth-log
atopacct           postfix
binfmt-support     procs
bluetooth          pulseaudio-enable-autospawn
console-setup.sh   rsync
cron               saned
crtmpserver        screen-cleanup
cubemap            speech-dispatcher
cups               spice-vdagent
dbus               sssd
gdm3               ubuntu-fan
grub-common        udev
hwclock.sh         ufw
irqbalance         unattended-upgrades
kerneloops         unsd
keyboard-setup.sh  uuid
kmod               vdr
lirc               whoopsie
lircmd             x11-common
matsstage@matsstage:~$ cat /etc/init.d/
23.10 0: bash 1.06 4x2.30h 7.60c4 100000 2024-
```

When using SysV, you have shell scripts in `/etc/init.d/` and you can add your own if you follow the rules.

slows down. As systems became bigger and needed more services running, administrators needed to manage services from boot and beyond. While scripting makes it possible, it is clumsy. In the beginning, it required bespoke shell scripts at every step. With *init* only starting 12 TTY sessions, the user had to start all services on their own. The first solution

## » WHAT ABOUT THE BSD?

As BSD has a longer history than Linux, it has an older initialisation system. It even predates SysV and has a similar structure to store files. A directory named `/etc/rc.d` contains the service files. They execute according to keywords embedded inside the files. Administrators must determine these dependencies when writing the scripts. This makes the

system easy to handle for administrators, which is why it is still popular among BSD users. It starts services the same way as SysV, using the path to the service script and the start parameter.

Despite this apparent simplicity, it requires exact scripts that determine all the actions to start and stop. Dependencies are also supported, but each and

every service you need must be in the file.

The BSD initialisation system is older than *Systemd*. Despite this, it works reliably and has been updated over the years. These updates make it a useful tool but not as encompassing as *Systemd*. Whether that is good or bad is a matter of opinion.

Given these differences, your choice depends on what

demands you have on the power of the system versus the power you can wield for different situations. With *Systemd*, apps and daemons already have support for the desktop. On servers, you have other systems with other needs. Finally, with embedded systems, you want to tweak everything to the last byte, in which case, you can get away with other solutions.



was to hardcode specific files where these scripts should go. That's the start of SysV's init and **init.rc** files.

Enthusiastic Linux users can find these files in **/etc/init.d**, **/etc/rc.d** or similar. At first, the files were pure shell scripts. Convenient if you are always writing shell scripts and educating people about them. Not so good for personal users or even companies. Solutions, in the '70s, were to create a special format for these scripts.

You find the version numbers of these systems in the name. Earlier Unix systems, called System [number], ending up at V (five.) Which leads us to talk about System V in this article. Unless you want to dig all the way down, SysV is the 'old' established system. Old-school hackers consider SysV adequate and simple to use as it depends on shell scripts. Parallel starts were not well supported, though developers had made efforts on this front.

Linux's first init system was rudimentary; its name is **simpleinit**. In this system, you would fill the **/etc/inittab** file with a simple table containing TTYs, the boot program and the value of **PATH** and **INIT\_PATH**. The boot process simply follows the **INIT\_PATH** and executes all scripts contained in the directory. If you have a **bootprog** value, it only runs that file the same happens when the **INIT\_PATH** is a file.

Seeing the way **simpleinit** works, you are up against another quirk: if you put **init=someprogram** in your boot options, 'someprogram' runs. You can use this to start even a regular app. Usually, this is a terrible idea as you'll drop the system's stabilising features. you can consider it when you build small embedded systems, using toolkits like *Buildroot*.

## Ubuntu the Upstart distro

In the first serious distribution, SLS (Softlanding Linux System), they deployed **simpleinit** as the default. For various reasons, other distros quickly superseded SLS. One was Slackware, which is still around.

Their init systems quickly moved to SysV, and mostly remained there until Ubuntu had other ideas. Canonical wrote a document to improve initialisation and supervision of services. Part of this effort is Upstart, no longer default in Ubuntu but available for enthusiasts (<https://wiki.ubuntu.com/ReplacementInit>).

Taking into consideration new, at the time, USB devices entering the market, the developers considered "events" as the basic concept for their new system. Event-driven systems define what any service needs before it starts. As an example, an SSH daemon waits for the event of networking running. As control lies with init, it controls what happens at all times.

At the time (2006-ish), the kernel developers added hot-plug features in a reaction to support for the aforementioned USB devices. The thinking then became focused on the events caused by changing hardware during runtime. Mounting disks that may or may not be there during boot is unreliable, making it necessary to wait for the event when it becomes available. Other issues that need this are network attached filesystems. Upstart, while popular, was short-lived in the history of init.

## The three philosophies:

- **Runlevels** run whatever matches the runlevel in lexical order; the filename decides which goes first.

- **Events** define an event for every service you want to start.

- **Targets** group services needed for each type of operation.

While Upstart development was going on, developers came up with other solutions. Both corporate entities (read Apple) and the BSD communities created other solutions. When it was time for a shakeup, Red Hat employees Lennart Poettering and Kay Sievers took inspiration from *Launchd* of Mac OS.

That inspiration started the *Systemd* effort. The main goal of the project is to unify the init process for all of Linux. In technical terms, the goal was to add parallelism in boot while also replacing different daemons. Some of the services *Systemd* can handle include network, devices and user management. You can also create your own services; the simplest is to create a *Systemd* timer instead of using *Cron* or *At*.

*Systemd*'s target roughly corresponds to runlevels, though it handles dependency instead. In *Systemd*, dependency decides which service must run before starting any other service. You use service files for control, they contain statements such as **After**, **Before** and **Wants**. These are all handling dependencies, not sequence. The main keyword regarding this is **target**.

A target is similar to runlevels; they change what type of system you run. On a desktop, that means whether you have a text-only system or a graphical desktop. When you run a server, a potential target is a web server and all that entails. For specialised cases, you write the target file yourself.

The developers at GUIX have their own ideas about the init system – they call it Shepherd. These ideas lead to differences, but not too many. The main one is that it can start user services and that you configure with the GNU GUILE programming language, an implementation of Scheme. Shepherd is still in version 0.10, so may not be fully stable and it may be lacking some features. However, if you install the GNU GUIX distribution, you get Shepherd as default.

## One init to rule them all

Chances are that you, reading this, are running *Systemd*. All major distributions default to using

### QUICK TIP

**Sinit** is the simplest you can choose. In 91 lines of code, it may be impossible to go smaller. You need to combine it with **Daemontools-encore**, at least according to the **Suckless project website**.

*Systemd* has more options to fill in and uses the INI file type; this is where controversy erupted during development.

```
WantedBy=suspend.target

matstage@matstage:~$ cat $(locate wifi-resume.service)
[Unit]
Description=Local system resume actions
After=suspend.target

[Service]
Type=oneshot
ExecStart=/bin/systemctl restart network-manager.service

[Install]
WantedBy=suspend.target

[Unit]
Description=Local system resume actions
After=suspend.target

[Service]
Type=oneshot
ExecStart=/bin/systemctl restart network-manager.service
```



## QUICK TIP

Are you adding your own services and wonder what will happen when you mess it up? Fear not – at least **runit** can run alongside your existing solution.

**Systemd.** The biggest reason for this is that it covers all bases. As you saw earlier, this also causes it to be big and complex. If you want to learn how to use it, you would be wise to try small experiments. Some changes risk the stability of your system, although **Systemd** is extremely robust.

To make experiments while staying safe, you can write scripts for any typical initialisation system and start it once. The simplest and safest way to this is to stop GRUB before boot and add `init=/sbin/init/` to your kernel options, changing the init string to your needs, of course. This makes your system boot using your alternative init file but doesn't change your next boot. A reboot clears all sins away.

## SysV scripts

The power of SysV is that you control it using shell scripts – this makes it simple to add your own services. All you need to learn are the basic ways SysV reports status and handles the state of services.

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: pulseaudio-enable-autospawn
# Required-Start: $local_fs
# Required-Stop: umountfs
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Enable pulseaudio autospawn
# Description: Enables autospawn for the pulseaudio daemon
### END INIT INFO

set -e

./lib/lsb/init-functions
```

```
case "$1" in
    start|reload|restart|force-reload)
        echo "autospawn=yes" > /run/pulseaudio-
enable-autospawn
        ;;
    stop|status)
        ;;
    *)
        ;;
esac
```

The header files contain an LSB-style stanza that controls the rules for the service. In the code above, you can see that it requires the local filesystem to run. Next you see how the service should react to runlevel changes. On the first lines of the actual script, it sets the environment variables and then loads any necessary functions.

Below, you see the case statement, which refers to the options you can give the service command when handling the service. For any service, you can add a script that runs for these occasions. If you are curious, any common distribution still leaves these scripts on your disk for backward compatibility.

To start a service, you would call it from its original position in the hierarchical filesystem. To run the above script, you would run `/etc/init.d/pulseaudio-enable-autospawn start`. As it calls the script, the case is **start** and the service starts. During boot, the script runs since it is in the correct directory.

## Upstart scripts

Upstart scripts do the same thing but try to be more compact. The focus is on dependencies, so you set the start and stop conditions. Runlevels are still a major part of the system, so you need to define those. What enthusiasts love about Upstart is that you can choose any conditions for starting a service.

In the code below, you can see that the Nginx server starts when the filesystem and the network are up. You could choose any other state to compare to, even your own homemade service. That service could ping a known server and start only after you know that is running.

```
# nginx
# description: nginx http daemon
# author: George Shammas <georgyo@gmail.com>

description "nginx http daemon"
author "George Shammas <georgyo@gmail.com>"

start on (filesystem and net-device-up IFACE!=lo)
stop on runlevel [!2345]

env DAEMON=/usr/sbin/nginx
env PID=/var/run/nginx.pid

pre-start script
# Perform pre-start actions
$DAEMON -t
if [ $? -ne 0 ]
then
    exit $?
fi
end script

exec $DAEMON
```

## » SMALL SYSTEMS – YOUR ROBOT!

It's common for small systems, like a robot, to use **BusyBox**. In this case, you can use **minirc**; this works just like **runit** but is even smaller. The developer has focused on making small binaries and a simple setup. Taking the init scripts from Arch Linux, they could create an extremely small system.

In this solution, a comma-separated variable denotes which daemons are configured. Another variable, **ENABLED**, contains a list of all services that start on boot. To configure new services or override the standard settings and services, you change a single configuration file: `/etc/minirc.conf`.

For even more granular control, you can support other **udev** solutions, making it possible to minimise memory or improve functionality by choosing between **mdev**, **Systemd**, **udev** and **eudev**, a fork of **udev**. If you need special scripts to run, the program runs `/etc/minirc.local`, where you can put your shell scripts. There are lots of examples on GitHub that you can use to create your own.

If you want to play with this, consider the difference between init scripts and service managers. Known service managers are **monit**, **perp**, **s6** and **supervisord**. It's common to choose from init scripts and service managers in conjunction. You can choose any combo, even use a combined binary (**runit** is one) and combine it with another service manager. To find out more, start with the GitHub page and search the Arch Linux wiki: <https://github.com/hut/minirc/>.



```
# Allow respawning of the NGINX process
respawn
respawn limit 10 5
```

The interesting part of this script is that you could use any conditions in the prestart script. Another neat feature is setting a respawn limit, in case things are repeatedly going wrong. Upstart, due to its simplicity, was popular for a time but as *Systemd* started rolling out, the mass of features won out, making *Systemd* the default in most distributions. Running Upstart scripts varies from SysV in that you must use the **service** command with the same options: `sudo service nginx start`. Most other actions work the same.

## Systemd scripts

They are not really called scripts in *Systemd*; rather, they're specifications for a service, unit and so on. The files use the INI format, with sections for what you are trying to achieve. The most common sections are **Unit**, **Service** and **Install**. There are a myriad of keywords to control the behaviour of each service or unit. Most of these keywords are easy to grasp if you understand the main idea of how *Systemd* keeps your system stable and efficient. In the below code, you can see an example of a service that runs in the background.

```
[Unit]
Description=EarnApp
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
Restart=always
RestartSec=5
ExecStart=/usr/bin/earnapp run
User=root
StandardOutput=null
StandardError=null

[Install]
WantedBy=multi-user.target
```

## Runit scripts

This system is extremely simple to use; the core of runit is `/etc/sv`. In this directory, you put a directory for each service with the script, always named `run`. Most scripts are less than five lines of code long. When you have those files, you link the directories to `/etc/service` to activate. The runit notices the new service within five seconds and starts it.

```
#!/bin/sh
TZ=MET-1METDST
export TZ
exec /usr/local/apache2/bin/httpd -DNO_DETACH
```

Look through the code – this is a long example. Two lines of settings are there to make sure the environment is correct for this service. When you have applications that need more parameters, you start it by adding a call to the config file, the way you would from the command line.

With start, stop and restart, it seems to do everything expected from an init solution. It can handle

dependencies, help with logs and even use runlevels, as in SysV and friends. If you want to try it out, you can actually run it together with *Systemd* and other service management software.

## Reasons to switch

For your ordinary desktop or laptop system, there is no real reason to leave – and plenty to keep – the default init system. However, those who do have intimate knowledge of the implications or want that knowledge may want to switch. The debate about how *Systemd* works is nitpicking over details that have the most impact for administrators.

While this debate is healthy, bringing new ideas and solutions to the Linux space, it does not help the ordinary user in the short term. On large servers where every improvement counts, it is useful to see how the system performs with the different options.

Equally, when you have a project that requires saving every byte for other jobs, you need to think about the init system. This is where you might switch to something simpler. Mainly, this can be beneficial if you choose a limited number of applications. You might want to just run monitoring software and display it. The number of packages plummet when your 'system' is as small as a credit card.

The story of init scripts in Unix and similar systems is fascinating but moves slowly. The reason is that they need to change only in response to other changes. A major change came with hot-plug functions in the kernel; next came the cgroups functionality. Both of these developments gave administrators a strong reason to also take advantage in the boot and management of their systems. As a regular user, you now have the advantage of this faster boot and robust response to new drives.

If you have the inclination to dig deep into how your system works, this is a fascinating subject. Be aware, though, that the system you are testing in will be unbootable at one point or another when you change such things. Be careful and have fun! No progress happens without failure – personal or for humanity. **LXF**

## QUICK TIP

**BusyBox** replaces the init daemon with itself, using a single file (`/etc/init.d/rcS`) for configuration. It has few features due to its small size – it is, after all, intended for embedded systems.

What makes *Systemd* seem so complicated is the wide array of files that link to each other to make things run.

```
dbus-fi.wl.wpa_supplicant1.service -> /lib/systemd/system/wpa_supplic[107/748]
ice
dbus-org.bluez.service -> /lib/systemd/system/bluetooth.service
dbus-org.freedesktop.Avahi.service -> /lib/systemd/system/avahi-daemon.servi
a
dbus-org.freedesktop.ModemManager1.service -> /lib/systemd/system/ModemManage
r.service
dbus-org.freedesktop.nm-dispatcher.service -> /lib/systemd/system/NetworkMana
ger-dispatcher.service
dbus-org.freedesktop.oaml.service -> /lib/systemd/system/systemd-oaml.service
dbus-org.freedesktop.resolve1.service -> /lib/systemd/system/systemd-resolved
.service
dbus-org.freedesktop.thermald.service -> /lib/systemd/system/thermald.service
dbus-org.freedesktop.timesync1.service -> /lib/systemd/system/systemd-timesyn
cd.service
default.target.wants
display-manager.service -> /lib/systemd/system/gdm3.service
display-manager.service.wants
docker.service -> /dev/null
docker.socket -> /dev/null
earnapp.service
earnapp.upgrader.service
emergency.target.wants
final.target.wants
getty.target.wants
graphical.target.wants
hibernate.target.wants
hide.me@.service -> /opt/hide.me/hide.me@.service
hybrid-sleep.target.wants
23.10 0:[tmux]* 1:bash# 38 8.17 4x2.5GHz 7.6657 109689 2024-02-13 10:27:30
```

» **MAKE STARTING WITH US EASY** Subscribe now at <http://bit.ly/LinuxFormat>

# BACK ISSUES » MISSED ONE?

## ISSUE 313

April 2024

Product code:  
**LXFDB0313**



### In the magazine

Discover how to use the ultimate hacker's toolkit, staying out of trouble while doing so. And join us as we take the Puppy Linux developer's new distro for a run and explore its container features. Plus, we have a *Roundup* of retro-gaming distros, an in-depth look at filesystems, tutorials on adding plug-ins to our LXF shell and getting more from your VMs, plus news, reviews and oodles more.

## ISSUE 312

March 2024

Product code:  
**LXFDB0312**



### In the magazine

Blast off into the future with a look at the five best next-gen distros, and discover whether Raspberry Pi or Orange Pi is the best SBC for you. Plus, learn how to rescue retro media, add NPCs to your own point-and-click adventure, emulate an analogue computer, and lots more. We've also squeezed in hardware and distro reviews, a password manager *Roundup*, news and more.

## ISSUE 311

February 2024

Product code:  
**LXFDB0311**



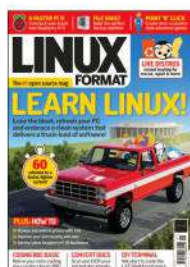
### In the magazine

Stay safe online by sending net nasties to *Pi-hole*, and read about the rise and fall (and rise again) of Basic, and the roll-out of the rolling-release Rhino Linux distro. You can also learn how to strengthen your shell history, set up a home CCTV system, add an inventory to your point-and-click adventure, and much more. Plus, check out our packed reviews, news and Raspberry Pi sections.

## ISSUE 310

January 2024

Product code:  
**LXFDB0310**



### In the magazine

Learn Linux with our in-depth guide to getting started with Elementary OS, or explore a more security-centric distro as we dive into Tails. Plus, find out how to perfect the art of OCR, finish your *WordPress* website, recreate the BBC Micro, and make your own point-and-click adventure. And no issue of *Linux Format* would be complete without news, reviews, Pi projects and much more besides.

## ISSUE 309

December 2023

Product code:  
**LXFDB0309**



### In the magazine

Manage your apps by harnessing the power of *Podman*, and discover how random numbers are used by the kernel to keep your data safe. Plus, find out how to master printing in Linux, customise your *WordPress* website, tweak your kernel scheduler, use Python to scrape the web, process your smart-home data, and much more. And make sure you read our *Roundup* of Raspberry Pi killers...

## ISSUE 308

November 2023

Product code:  
**LXFDB0308**



### In the magazine

Bring together all your movie, TV, music, audiobook and photo streaming needs under one roof. Plus, we take an in-depth look at the Pop!\_OS distro, explain how networking works, and guide you through tutorials on *WordPress*, fast file management, reviving the BBC Domesday Project and more. And don't forget to check out our packed reviews section, plus we examine the brand-new Raspberry Pi 5.

To order, visit [www.magazinesdirect.com](http://www.magazinesdirect.com)

Select **Single Issues** from the tab menu, then select **Linux Format**.

Or call the back issues hotline on **0330 333 1113**

or **+44 (0)330 333 1113** for overseas orders.

Quote the product code shown above and have your credit or debit card details ready.



USA? EU? THE MOON\*?

UK readers  
turn to  
p16

# SUBSCRIBE!

Don't wait for the latest issue to reach your local store – subscribe today and let *Linux Format* fly straight to you. Faster, cheaper and with DRM-free archive access!



2 GREAT  
WAYS TO  
SUBSCRIBE

Digital-only and  
print+digital  
bundles!

» USA  
From \$135.49  
For 13 issues

» REST OF THE WORLD  
From \$135.49  
For 13 issues

» EUROPE  
From €119.49  
For 13 issues

## IT'S EASY TO SUBSCRIBE!

Visit [www.magazinesdirect.com/linux-format](http://www.magazinesdirect.com/linux-format)

Call +44 0330 333 1113

Lines open Monday-Friday, 9am-5pm, UK time.

\*We don't actually deliver to the Moon. Yet.

## SUNSHINE &amp; MOONLIGHT

Credit: <https://moonlight-stream.org>, <https://github.com/LizardByte>

# Ultra-smooth home game streaming

Promising the Sun and the Moon, **Michael Reed**'s heavenly bodies promise perfect gaming streaming on most PC hardware.



**OUR  
EXPERT**

**Michael Reed** is a consummate Linux professional who has been moonlighting for us so that he can live in the sunshine. (are we actually paying for these gags?—Ed)

## QUICK TIP

You can install the *Sunshine* server on to a VM to control it. We found that we had to enable 3D acceleration before it would work in *VirtualBox* for simple desktop sharing. Performance was good.

**S**unshine and Moonlight are open source server and client programs that allow desktop sharing and game streaming across a network. Originally, both were a reimplement of an Nvidia protocol, but in their current state, they don't require any Nvidia hardware or software to run. Having said that, they can take advantage of GPU hardware for video encoding, if you have it.

You can also share games and desktops between different platforms and operating systems. The platform support for the client side app is even more extensive and includes the Raspberry Pi along with mobile and smart TV-type devices, as well as regular Linux, Windows and Mac computers. What Moonlight and Sunshine offer together is completely free, highly performant and extremely easy to set up and use.

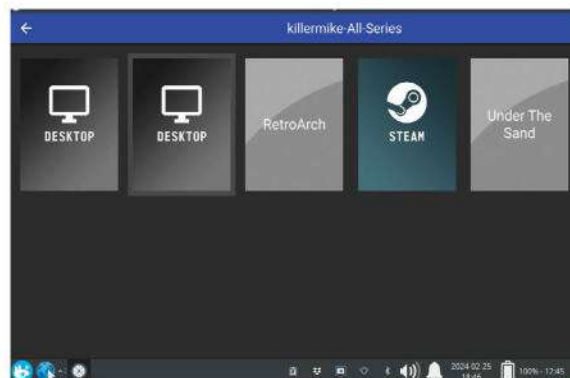
Whether you are gaming or running desktop apps, this is how it works: as you sit using Moonlight on the client computer, the input from your keyboard, mouse and gamepad are sent through the network to the computer running Sunshine. Sunshine uses extremely efficient audio and graphics compression codecs to send the screen and the audio of the computer back to the Moonlight program, so that you feel as though you are running the software locally on the client computer.

How you connect everything is up to you. For example, you could run Sunshine in a virtual machine and run Moonlight on a smart TV to give you a Linux desktop on your TV. You could take your laptop into the next room and have access to your desktop PC. The experience in terms of latency and visual quality depends on the power of your server computer and speed of your network. With a few compromises in that area, you can even use the setup over the internet – handy if you need the power of your main computer on your tablet or phone when you're out and about.

## Moonlighting clients

Sunshine (the server) and Moonlight (the client) are installed separately as they are different open source projects that use the same protocol.

The Moonlight website (<https://moonlight-stream.org>) sells the project a bit short because it doesn't make it clear that the client can operate without the proprietary Nvidia software on the server end. We'd



Moonlight's launchers can be created from scratch and configured to launch specific programs on the host machine.

have preferred it if the site bigged up the open source, multiplatform side of things, because there is an open source server in the form of Sunshine, which implements the GameStream protocol and works perfectly with Moonlight. It's also the case that you don't need any Nvidia hardware on either the client or the server machine. If you do have a supported Nvidia, AMD or Intel GPU, the software can take advantage of it, however. If you don't, software encoding is perfectly viable if your CPU is reasonably powerful.

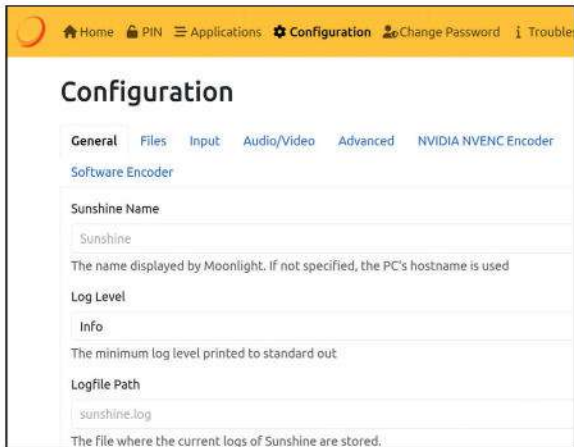
We were impressed with the installation options provided by the Moonlight project. For typical desktop Linux, the Snap or Flatpak options might be the easiest to work with. You might also be interested in the AppImage package as it requires no installation at all. For our testing purposes, we began by installing the client on to a low-powered laptop with the following command: `snap install moonlight`

## Serving up the Sunshine

Once you have Moonlight installed, it's time to install Sunshine, the server. Typically, you would install this on another computer on your network. The Sunshine website (<https://app.lizardbyte.dev/Sunshine/>) has various DEB and RPM packages for different Linux distros and platforms in addition to Flatpak packages.

There are some downsides to the Flatpak version we tested. First of all, it pulls through more than 1GB of files to complete the installation, quite a bit more than





Sunshine is configured using its web interface. Note that you don't have to be logged into this to connect to Sunshine from Moonlight.

a DEB installation. More importantly, Flatpak uses sandboxing to lock off system resources from a program that uses it. In the case of *Sunshine*, we found that many of the examples from the documentation wouldn't work when we tried to expand the system beyond basic desktop sharing. We could run Steam over desktop sharing like any other application, but we couldn't launch Steam directly from *Moonlight*.

Flatpak is worth considering if you simply want desktop sharing and an easier installation experience. Go for DEB or RPM if you want to be able to expand the system in areas such as directly launching software and other things that require access to the overall system, such as directly launching emulators.

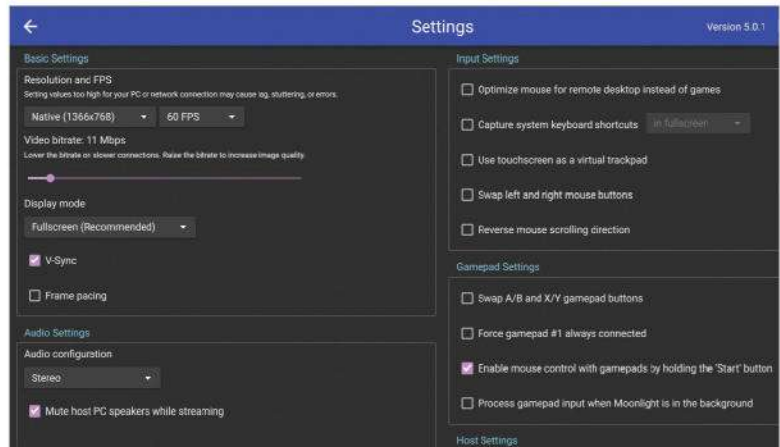
We installed the DEB version to Linux Mint and Ubuntu-based systems, to explore everything the system was capable of. This wasn't completely straightforward because we had to manually add some dependencies (libbmx1, libboost-log1.74.0, libboost-program-options1.74.0). If you go this route, *Sunshine* can tell you if there is an updated version available, and you have to fetch and install it manually.

Once you have installed *Sunshine*, it should be runnable from the standard desktop launcher. When you do this, the server configuration and management tool opens in a browser tab, where you can enter a password and username. You can launch *Sunshine* from the command line, which causes it to spew forth some useful logging information. However, when the web interface is running, it has a Troubleshooting tab with a fully searchable logging section.

See the official documentation (accessible through the <https://github.com/LizardByte/Sunshine> site) if you want to autostart *Sunshine* on your computer. *Sunshine* is actually implemented as a service, and you have to copy the example from the docs. Having configured this, as per the docs, you can start *Sunshine* with `systemctl --user start sunshine` and enable automatic startup with `systemctl --user enable sunshine`. However, it's better to manually launch *Sunshine* while you're still testing it and setting it up.

## Moon and the server

We found that, across different machines and configurations, *Sunshine* and *Moonlight* have a good chance of working adequately with the initial configuration. Speaking of configuration, once



Moonlight's configuration options. Our first port of call is usually to match the resolution with that of the device we're using.

*Sunshine* is running, its configuration pages should be accessible at <https://localhost:47990/>. You can also get to this page by selecting the Open Sunshine option when right-clicking on the status area icon. There is a small wrinkle at this stage because most browsers complain that the site is not secure. This is because, at the time of writing, the *Sunshine* server does not produce a valid SSL certificate. However, as this is a server running on your own computer on your own network, it is safe to ignore the warning and continue to the configuration web page.

With what you have so far, you should be able to connect to the *Sunshine* server on one machine with the *Moonlight* client on the other, so you can view and control the desktop of the other computer.

Start by discovering the IP address of the *Sunshine* server. How you do this depends on your distribution, but what works on nearly all Linux distros is to issue the command `ip a`. This is the line of interest on a Linux Mint computer:

```
inet 192.168.0.22/24 brd 192.168.0.255 scope global
dynamic noprefixroute eno1
```

Typically, you're looking for a number in the form of **192.168.x.y**. The first variable part, **x**, is normally either 1 or 0; **y** is the number that the computer has been assigned on the network. The first part of the IP address tells network devices that this is an address on your internal network rather than out on the internet. There are other schemes but most routers default to **192.168** for this part. In the example above, the complete IP address for the server computer is **192.168.0.22**.

## Orbital pair

Run *Moonlight* on the other computer. You're greeted with an empty machine selection screen. Add a new server by clicking on the + icon on the bar at the top of



Moonlight's statistics overlay is a useful tool to work out how good your setup actually is. It's invoked with Ctrl+Shift+Alt+S.

## QUICK TIP

Adding *Moonlight* to an Amazon Firestick simply involved searching for the app in the Applications menu. It worked well on a TV, but it's the user's job to connect the needed input devices.

## QUICK TIP

The Android version of *Moonlight* (download from Google Play Store) is difficult to use on a phone because of the small screen, and is more suited to tablets, handhelds and Chromebooks.

the window. When prompted, enter the IP address of the server that you have just obtained.

Next, you are prompted to pair the computers. This is a security feature so random computers on the network can't suddenly take control of each other. The computer running *Moonlight* displays a PIN. Enter this into the *Sunshine* server by going to the web config page and selecting the PIN tab. Apologies if you have to jog upstairs to enter this into your other computer.

This takes you back to the Computer selection screen. Click on the big icon for the server you've just added. This takes you to another set of icons (launchers). We'll get into more detail about configuring these launchers and creating custom ones later on.

## Staring at the sun

Select the first launcher (Desktop) to make a regular desktop connection. If this works properly, you should now be looking at the desktop of the other computer (including sound). Sure enough, if you move the mouse, it should move the mouse pointer on the *Sunshine* server desktop. It's a good idea at this point to launch a reasonably complex app such as a web browser on the server to fully test that everything's working properly.

To navigate while using *Moonlight*, you need to learn a few keyboard commands to control the server. These are a bit longer than the ones you might be used to as



Running *Moonlight* on an Android phone. This version of *Moonlight* can pass mouse movements and overlay a game controller.

the developers couldn't risk choosing combinations that are already defined on the server or client machines. Thankfully, they are easy to remember because they all begin with Ctrl+Shift+Alt. So, for example, Ctrl+Shift+Alt+x switches the client window between full-screen and windowed mode. In windowed mode, Alt+Tab works on the computer you're using in the normal way. So, remember this one keyboard combo, if none of the others, so that you can use it to switch away from the *Moonlight* window.

Ctrl+Shift+Alt+q exits the sharing connection, returning you to *Moonlight*'s launcher icons. Something interesting happens when you do this because the desktop of the server machine is still running. So, you could exit that connection and then immediately switch to another machine, which could be a desktop, a laptop or a mobile device. If you're running a game, don't forget to pause it while you move between computers.

There are options within the *Moonlight* and *Sunshine* configuration sections to control how the super key and other key combinations are handled in terms of what is passed to the host machine.

## Moon launch!

So far, we've been using the first launcher (Desktop), but let's take a look at what these launchers can do. Once you double-click on a computer in *Moonlight*'s window, you are taken to a selection of launchers, the first of which is Desktop. What each of these launchers does is establish the connection between the client and server computers. Then, the configured actions are carried out. These actions are defined using *Sunshine*'s configuration. This is the best scheme because it means that all machines that can access a particular *Sunshine* server have access to the same launchers, meaning far less setup work.

For convenience, we created a launcher to launch the driving/exploration game *Under the Sand*, a game we had installed on Steam, and it serves as a good, typical example of directly launching a Steam game. Launchers are created by visiting the Applications tab in the *Sunshine* configuration page (right-click on the *Sunshine* icon in the control panel and select Open *Sunshine* or visit <https://localhost:47990/>).

Select the +Add New option. Add *Under the Sand* in the Application Name field. For various reasons, a Steam game has to be added as a detached command. To get the command we needed, we typed the name of the game into the application launcher of our desktop and then right-clicked to edit that entry. It turned out that the command that launches that game was `steam://rungameid/1062960`. We cut and pasted this

## » MOBILE MOONLIGHT

Controlling a fairly powerful desktop PC running Linux Mint from a repurposed Chromebook running Ubuntu (Intel N4500, two cores @ 2.8GHz) over Wi-Fi offered a great experience for desktop apps.

Initially, the picture wasn't completely clear because the screens were different sizes and the resolutions didn't match. We improved the clarity by setting the resolution in *Moonlight*'s settings to the native resolution of the laptop (1,366x768). To scale the interface up and improve clarity further, we changed the resolution of the server machine, using its display settings. *Moonlight* and *Sunshine* handled that without dropping the connection.

We'd have been happy using this to do word processing or browse the web. Some things were faster than we'd expect from that laptop, as the host machine had more processing power and memory. A few things that required a lot of screen updates were less fluid, though.

For gaming we were able to get nearly 60 FPS at an acceptable quality. There was some slight latency, but depending on the style of game and how sensitive you are to such things it might be fine.



Running *Moonlight* on a laptop means that you can take your main desktop with you.



into the Detached Command field and pressed the + icon to add it. From this point on, we were able to launch *Under the Sand* directly in the launcher.

We found that the option to launch Steam in Big Picture mode (full-screen for TVs) didn't work the way we wanted. We edited that launcher (in the Application tab of *Sunshine's* configuration), deleted the existing Detached Command entry and replaced it with `/usr/bin/steam steam://open/gamepadui`. We set Big Picture mode to default to full-screen in Steam itself.

## Lighting up emulators

The server machine had the emulator front-end *RetroArch* installed, and we fancied being able to launch it directly. To do this, we created a new entry with *RetroArch* in the Application Name field. As before, we discovered the launch string from the application launcher:

```
/usr/bin/flatpak run --branch=stable --arch=x86_64
--command=retroarch org.libretro.RetroArch
```

This time, we cut and pasted it into the Command field. Sure enough, it worked as expected and the game controller was transmitted over the network connection properly. This gave us a setup where all we had to do was select the *RetroArch* icon on any device with *Moonlight* installed to cause *RetroArch* to open up full-screen on our device. Of course, in reality, it was running on our desktop computer.

However, we had to redefine this controller layout in *RetroArch* as the axes and buttons are translated differently compared to using the controller directly on that machine. As the launch command was added to the Command field rather than Detached Command, when we exited this game, *RetroArch* was closed down on the host machine correctly. Note that *Moonlight* refers to all virtual connections that are reached via the launcher screen as games even if it's a desktop share or similar.

It's possible to get a bit more complicated as *Sunshine* enables you to add chains of commands or commands that are called when the 'game' is exited. Basically, if you can start or stop something on the host machine by running commands, you should be able to get it running through *Sunshine* and *Moonlight*.

## Shining online

The *Sunshine* server comes with Universal Plug and Play (UPnP) support built in. This can carry out the port forwarding setup on your router to connect an application to the internet automatically. However, we had difficulty getting this working, and searching around online, we found that we weren't alone. What did work was setting up the port forwarding ourselves. To do this, we first switched off UPnP support (Configuration > General tab in *Sunshine's* settings).

We found the correct port settings in the Advanced tab of *Sunshine's* configuration, where they are listed. The snag is that this means setting up five port forwarding rules in the router. See the instructions for your broadband router for how to set up port forwarding. However, once we did it, sure enough, we were able to add our *Sunshine* server by deleting it

## » BAKING Pi WITH MOONLIGHT

To install *Moonlight* on a Pi (a Pi 400 running Raspberry Pi OS with a wired Ethernet connection), we followed the instructions on the website, which involved first updating the firmware of the Pi and then copying and pasting a line of code into a command terminal.

This gave us a version of *Moonlight* that looked identical to that on other platforms.

For desktop sharing, the performance was very good, and there were times when we could almost begin to forget that we were using a remote connection. As ever, setting the resolution to the native resolution increased the crispness of the screen. The black on white text in a word processor, for instance, looked perfectly crisp.

Things weren't as impressive when it came to gaming. The setup was usable once we had limited the frame rate to 30fps and set the bitrate to 20Mb/s, but it felt like a poverty setup for fast action games. When we switched to strategy game *Frostpunk*, things improved and the detail of the graphics and text was excellent.



A Moonlight/Sunshine server can give you access to applications that wouldn't normally be viable on a Pi.

from *Moonlight* and re-adding it using the external IP address of our router. We obtained this IP address by searching Google for a site that showed our external IP address (<https://nordvpn.com/what-is-my-ip/> in our case). We were using quite a fast fibre connection, but we were pleasantly surprised by the performance, particularly for desktop sharing.

As with any type of server, you have to use your own judgement as to the safety of setting it up for online use via the internet as opposed to using it on your local network. We couldn't see any problems with the arrangement because before a device can connect to your *Sunshine* server, the user would have to manually add a four-figure PIN to the PIN tab in the *Sunshine* server. By default, this can't be accessed from machines outside of your home or office network. As with all server software, if there were a security flaw that hackers could exploit, it's possible that they could get some access to your network and devices. Checking around online, it looks like *Sunshine* is a popular piece of open source software, and we couldn't find any complaints about its security features. **LXF**

IP address	Port range	Port range	Protocol	Enabled
192.168.0.22	47984-47984	47984-47984	TCP	✓
192.168.0.22	47989-47989	47989-47989	TCP	✓
192.168.0.22	47990-47990	47990-47990	TCP	✓
192.168.0.22	48010-48010	48010-48010	TCP	✓
192.168.0.22	47998-48000	47998-48000	UDP	✓

To make *Sunshine* available over the internet, look up the port forwarding instructions for your brand and model of router.

» **STREAM US TO YOUR DOORSTEP!** Subscribe now at <http://bit.ly/LinuxFormat>

## EMULATION

# Reviving the MK14 retro home computer

**Mike Bedford** uses emulation to learn about Sinclair's first sub-£100 computer, which predated the better known ZX80 by three years.



## OUR EXPERT

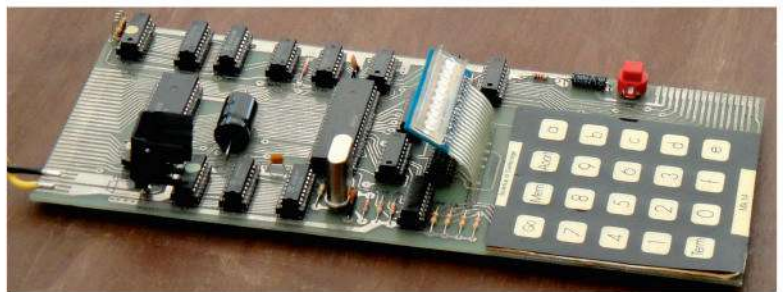
**Mike Bedford** hadn't even thought he might one day own a computer by the time the MK14 appeared. That makes it no less of an interesting diversion for him, though.

**T**he Sinclair ZX80 went on sale in 1980 at £99.95, or £79.95 in kit form. In so doing, it was responsible, in no small part, for kick-starting the British home computing revolution of the '80s. However, it doesn't tell the whole story. Nor is it correct to say, as is often suggested, that the ZX80 was the first computer to be offered for less than £100. That honour goes to a much lesser-known British computer called the MK14, our subject here. It's probably quite different in many ways from anything you've seen before, and it was nothing like that better-known home computer we referred to above. But it's no less a part of our computer heritage, so we thought it would be interesting and informative to delve into its unique features. This isn't just talk, though, so if you fancy getting some hands-on experience of the MK14, we show you how to do that via emulation.

## Introducing the MK14

Launched in 1977, the MK14 cost £39.95, making it half the price of the ZX80 in kit form. This is an entirely appropriate comparison, though, because our quoted price of the MK14 is also for a self-assembly kit. In fact, the MK14 was only ever made available as a kit of 31 parts, including the circuit board, all of which had to be soldered together. The company responsible for the MK14 was Science of Cambridge who, in the fullness of time, became Sinclair Research of ZX80, ZX81 and ZX Spectrum fame.

The specifications of computers of the '70s and the early '80s tend to be eye-openers because of the almost unbelievably small memory capacities and processor speeds. The MK14 was no exception, having 256 bytes of RAM and 512 bytes of ROM. This non-volatile memory held the monitor program, which allowed programs to be entered and executed, and the results displayed. Where the MK14 differed from almost all other microcomputers of the time, though, was its processor. The soon-to-be-launched home computers of the '80s virtually all had Z80s or 6502s,



It might have been the first sub-£50 computer, but the MK14 was also one of the most basic.

and some earlier US machines had Intel 8080s. Not the MK14, though, which had a SC/MP. Pronounced Scamp, this little-known processor from National Semiconductor was an 8-bit design, which was clocked at 4.4MHz on the MK14.

While this figure is similar if not higher than the clock speeds of most of the early '80s home computers, a clock figure is a notoriously poor measure of performance. Indeed, the SC/MP compared unfavourably to many of its competitors – see box (*opposite*) for more details. We should also point out that the MK14 didn't have a case; it was purely a circuit board. In addition, it didn't include the necessary 5V power supply, which you'd have needed to buy separately – just like a Raspberry Pi!

Input and output on the MK14 were basic – any expectations of a keyboard and video display would have led to disappointment. Input was via a 20-key pad with keys for 0-9 and a-f for hexadecimal entry, plus various functions. Reports indicate that the keypad was dreadful from an ergonomic viewpoint. Output was to a tiny eight-digit seven-segment LED display capable of displaying hexadecimal numbers.

Expansion was available, but limited. On-board expansion allowed the RAM capacity to be increased to 640 bytes and an I/O device to be added, thereby giving 16 I/O pins. A few off-board expansion options were also available. An add-on memory board allowed the RAM capacity to be increased by 4K, and the ROM capacity by 16K. A cassette interface allowed programs to be stored on an audio cassette tape, although the speed was reportedly pedestrian at an estimated four characters per second. And finally, an

## QUICK TIP

A version of Tiny BASIC has been written for the SC/MP – it's called NIBL. So, we might wonder if it can run on a suitably expanded MK14, with memory expansion and VDU cards.



expansion card, which cost almost as much as the MK14, permitted a VDU (visual display unit) to be used as an alternative to the seven-segment LED display. It could display 16x32-character text with only upper-case letters, or 64x64 pixel graphics, all in monochrome. It output a signal to a TV.

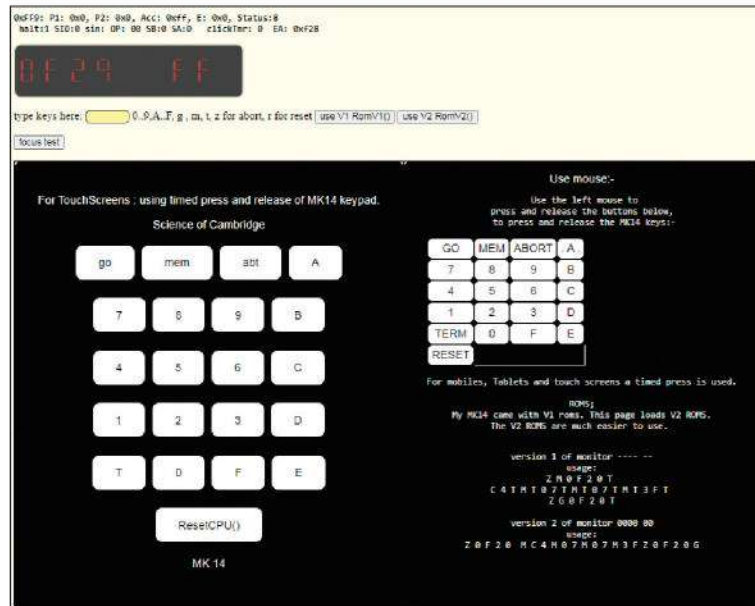
Despite this lacklustre specification – at least by today's standards, or even the standards of the early '80s – the MK14 was a big seller at a time when home computers were unheard of. Certainly, it didn't sell nearly as well as the later Sinclair computers, but we can't help but feel that its 15,000 reported sales were a harbinger of things to come.

## Emulator options

We found two options for emulating the MK14 in software and online. A software solution – at least in theory – is the open source **ZEsarUX** (<https://github.com/bernandezba/zesarux>), which is available for Linux. Generally thought of as an emulator of the classic Sinclair machines, it also includes support for the MK14, but not in our experience and there's no documentation to help.

Instead, an online emulator would seem entirely appropriate for your brief foray into programming the MK14. And the good news is that we found one that does the job admirably at [www.dougrice.plus.com/dev/seg\\_mk14.htm](http://www.dougrice.plus.com/dev/seg_mk14.htm). We did have to carry out a bit of detective work on the user interface to get the best from it, though, so here's what we discovered to give you a helping hand.

First of all, by default two versions of the 8-digit seven-segment display are shown, the upper one having the red segments on a black background, the lower one on a white background. The status of these are controlled by the two checkboxes labelled ASCII and SVG against Display Using. Initially, only SVG is checked and only the lower display reflects what would have appeared on the MK14's display. If you also select ASCII, then the upper display also starts to reflect the



This online emulator can give you an inkling of what it was like to program an MK14, laborious entry of hexadecimal numbers and all.

MK14's display. And if you deselect ASCII, then the lower display disappears.

If you want something as close as possible to the appearance of the MK14, the ASCII display would be your best choice. However, if the SVG version is displayed, you also get several status LEDs. These weren't available on the MK14, so you might consider it cheating to use them, but they could make debugging easier. If you do decide to ignore them, though, this status information was written to various memory locations when execution returned to the monitor program. This is described in the original MK14 *Training Manual*, which you can find at <https://bit.ly/lxf314manual>. Also on the subject of documentation, another useful resource is National Semiconductor's *SC/MP Programming and Assembler Manual*, which is available from <https://bit.ly/lxf314scmp>.

Next up is the keypad and, again, there are two variants, arranged side by side. The left-hand one claims to be for touchscreen use, and the right one for use with a mouse. However, while we didn't try the emulator with a touchscreen display, we found that both worked with a mouse. And if two options are good, surely three are better. And that third option involves using keys on your keyboard, but that only

## QUICK TIP

The MK14 wasn't the only minimalist computer based on the SC/MP. In 1976, Bywood Electronics released the £56 SCRUMPI kit. Its user interference comprised just toggle switches and single (not seven-segment) LEDs.

## » NATIONAL SEMICONDUCTOR SC/MP

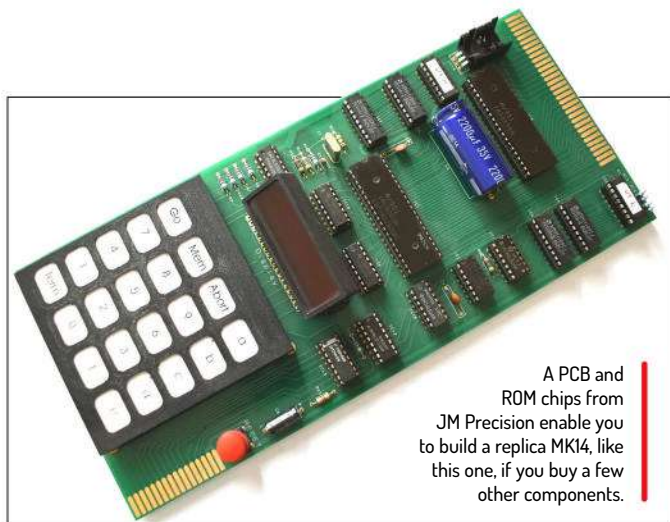
Back in the late '70s, Intel and Motorola were the big boys in the microprocessor market, with Zilog (Z80) and MOS Technology (6502) catching up. But those were the days of diversity. Processors were produced by countless companies, some forgotten and most others no longer in that market. One such company, now defunct, was National Semiconductor, and

the SC/MP was its first single-chip microprocessor.

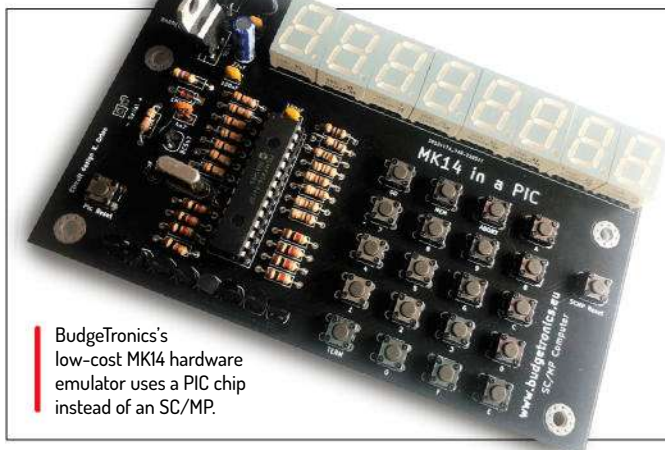
Also known as the INS8060 and INS8070 in the second and third generations of the chip, SC/MP was an 8-bit design. Nothing new there for the '70s, but the SC/MP was designed to be a low-cost chip. In fact, those curious mnemonics stand for Simple Cost-effective MicroProcessor, although we

couldn't discover just what the SC/MP cost. However, since the entire MK14 kit cost only £39.95, it must have been significantly less than the Intel 8080 or Motorola 6800, both of which had a retail price of \$360. What we can say with more certainty, though, was that to achieve its low cost, the SC/MP had a bit-serial ALU (arithmetic and logic unit). By way of contrast,

competing microprocessors employed parallel designs for their chips' ALUs. The upshot of this was that the SC/MP was slower than most other 8-bit chips of the era at the same clock speed. One source puts the 4.4MHz MK14 capable of 0.2 MIPS. A similar speed 6502 would touch 1.9MIPS the 68000 was capable of 1.4MIPS but was a full 16-bit CISC.



A PCB and ROM chips from JM Precision enable you to build a replica MK14, like this one, if you buy a few other components.



BudgetTronics's low-cost MK14 hardware emulator uses a PIC chip instead of an SC/MP.

works if you first click in the yellow box against Type Keys Here. Note that, in addition to the 20 keypad keys, there appears to be a 21st key labelled RESET. This actually emulates the MK14's reset button, which was a pushbutton separate from the keypad.

## SC/MP assembler

So, now you know your way around the emulator, it's time to try your hand at entering and executing some code, and perhaps graduate to writing your own code. Given the MK14's modest specification, and the limitations imposed by that hexadecimal keypad, you probably won't be surprised in the least to learn that Basic wasn't an option. Instead, programming was at a much lower level. It wasn't even possible to enter code in assembly language—instead, it had to be programmed in machine code. This meant that you had to think in machine code or, alternatively, you had to assemble your assembly language source code by hand, a notoriously error-prone process. Depending on how much you hanker after the true MK14 experience, you might think you could miss all this out by using an assembler, albeit not on the emulated MK14. We didn't manage to find one, though, at least not one for Linux, so there's a possible project for you.

As a first exercise, we suggest you enter and execute the first program shown on the emulator's web page, and it's trivially simple. Here's the code, as it would appear in SC/MP assembly language:

```
START: LDI 0AA
XRI 55
ST 2
XPPC P3
```

To start, a couple of introductory comments. First, we've given the first instruction a label, specifically **START**, even though it's not really needed because it's not referred to by any other instruction. However, such labels will be needed in more complicated programs, so we've added it here so you can recognise labels in the later programs. And second, the **XPPC** instruction is followed by **P3**, which is the syntax used in the SC/MP manual, but the MK14 manual uses just **3** instead. We're not sure whether both are valid or whether different syntax was used in different assemblers. Since you're probably not going to be using as assembler, though, it's a moot point.

Now to the instructions. The first instruction is **LDI**, which stands for Load Immediate. Its purpose is to load the value stored in the byte after the instruction's

op-code – that is, **AA** hexadecimal – into the processor's accumulator. Next comes **XRI**, which stands for XOR Immediate, XOR being exclusive OR. It XORs the value in the byte after the op-code – that is, **55** – with the value in the accumulator, the result being stored in the accumulator.

On to the **ST** instruction, which stands for Store. Its purpose is to store the value in the accumulator to the memory location that is obtained by adding the contents of the next byte, in this case **2**, to the program counter. Since the next and final instruction is just a single byte long, this causes the value in the accumulator, which is the result of the program, to appear immediately after the program code. And finally we have that final instruction, **XPPC**, which exchanges the contents of the program counter with that of a specified pointer, in this case pointer **3**. Even though the SC/MP didn't have an instruction to call and return from a subroutine, it could be implemented in software. And, on the MK14, the **XPPC 3** instruction caused control to return to the monitor program.

And finally, before you enter and run the code, let's follow it through by hand to see what you should expect to see. After the **LDI** instruction has executed, the accumulator contains **AA**. The **XRI** instruction then XORs that **AA** in the accumulator with **55**. If we point out that the binary for **AA** is **10101010** and the binary for **55** is **01010101**, it should be clear that, after this instruction, the accumulator will contain **FF** – that is, **11111111**. The web page describes exactly what keys you need to press to enter that code, starting at address **0F22**, and to execute it. After it's executed, the monitor displays the address and contents of the location after the end of the program. What you should see, therefore, is **0F29 FF**. The key sequence to enter and execute the code, as shown on the web page, is fairly self-explanatory.

There are additional and more complicated programs on the web page to try, but the source code isn't shown for most of them. Nevertheless, while it's still not too much more complicated, you might like to try the second example, which writes directly to the segments of the final digit of the seven-segment display. And if you can forgo the experience of entering endless sequences of two hexadecimal digits followed by the mem key, there's an easier way to enter and execute the code. Just click on the Setup button below the program listing to enter the code into memory, and then click on Go. Running the second example causes

## QUICK TIP

The memory capacities of early micro-computers, like the MK14's 256 bytes, will be quite an eye-opener to today's programmers. After all, that's just a quarter of a billionth of a gigabyte.



the final digit to show a pattern, as defined by the program, which doesn't correspond to any hexadecimal digit. Note that the program doesn't cause control to return to the monitor program because that would immediately cause the entire eight-digit displayed to be overwritten. So, to return to the monitor you have to press Reset.

If you want to try other examples, there are several in the MK14 manual, but they do tend to be quite long. To get to grips with SC/MP programming, therefore, you might be better devising your own examples, so you can start with something simple and build up. If you're lacking inspiration, how about building on our previous sample code to do more than just XOR? You could, for example, also carry out OR and AND functions, and even some simple arithmetic, writing the results to several consecutive memory locations before exiting. We recommend, though, that you change the two values of AA and 55, these numbers being particularly uninspired, since the two values XORed give the same result as if they're ORed.

The precise way in which the contents of memory locations are inspected and edited – both of which you need to do – are perhaps not obvious from what we've seen so far. So, consult the MK14 manual first.

### Hardware options

The online emulator we've seen so far seems to emulate the inner workings of the MK14 just like the original, and the graphical representation of the keypad and display are good enough. But if you want the real retro experience, nothing can beat using some real hardware. You could buy second-hand MK14s, but they're rare and expensive. We looked online and couldn't find any for sale, but reports suggest that, if you can find one, you'd pay between £500 and £700. That's surely only for the serious collector. There are lower-cost options, though, as we're about to see.

First up, we note that bare PCBs from which you could build your own MK14 are available for around £25. The potential snag, though, is getting hold of the obsolete parts to solder on to these circuit boards, most notably the SC/MP, the ROM containing the monitor program, and the 2112 RAM chips. However, this doesn't seem to be as much of a problem as we'd feared. A PCB that's an exact replica of the original MK14 circuit board is available from <http://jprecision.co.uk>. Also available from the same source is a pair of ROM chips, pre-programmed with the monitor code, and they say that all the other parts are available. Indeed, we can confirm that SC/MPs (in the guise of INS8060) and 2112s are available. If you shop around, we estimate that you could build a working MK14 in this way for not much more than £150. And it just so happens that, in real terms, this is about £70 cheaper than it would have cost to buy a real MK14 back in 1977.

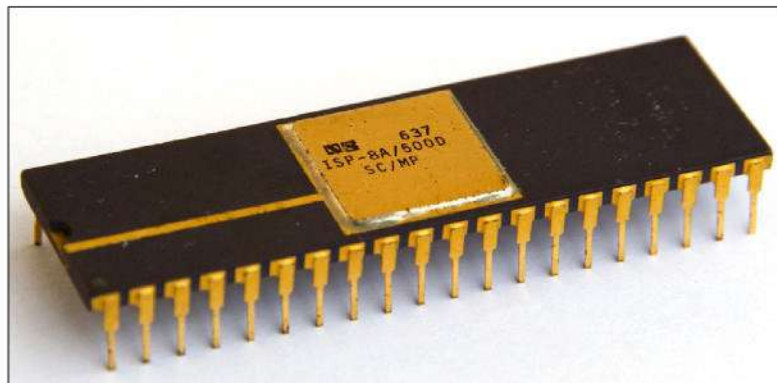
## » DEVELOPMENT KITS

The MK14 was intended as an enthusiasts' computer, even though it seems to have more in common with a development kit, aka an evaluation kit. Indeed, SC/MP manufacturer National Semiconductor sold a development kit called the SC/MP Introkit, which was very similar to the MK14. So, what do we mean by a development kit?

These kits are sold by semiconductor manufacturers to allow engineers to get some experience of their products. This will, it's hoped, pave the way to companies building their own designs using the core processor or microcontroller. Ease of use didn't have the same importance as with traditional home computers, and these kits often commanded a high price tag. These two facts don't altogether paint a good picture, so why might you be interested in development kits? We'll give one example that might just persuade you to investigate.

PIC microcontrollers from Microchip are cheap and widely used in industry. They also have a following among enthusiasts. But you're not going to find a 'proper' computer based around a PIC chip. So, either you have to jump straight into building your own design, or first get some experience using a development kit. Fortunately, the high price tag doesn't always apply. To quote Microchip, "The PIC32MM Curiosity Development Board is a demonstration, development and experimentation platform based on the PIC32MM0064GPL036 low-power, low-cost microcontroller. The board has a built-in programmer/debugger and provides all of the hardware necessary to get started developing a complete embedded application." This board and similar for other PIC chips costs from around £30.

A cheaper and more practical solution, albeit less authentic, is to build a single-board computer that looks like an MK14, or at least has a similar keypad and display, but uses different components. Indeed, these have been built, and some are available for sale. Typically, the SC/MP is replaced by a more modern microcontroller that's programmed to emulate the SC/MP's instructions and the operation of the MK14's monitor program. One such product is provided by BudgeTronics ([www.budgetronics.eu](http://www.budgetronics.eu)) and it costs just €45. It uses a PIC microcontroller, so the board doesn't look much like an MK14 of old but, crucially, it has a similar-looking keypad, and it has an eight-digit seven-segment LED display. And what it lacks in not looking like an MK14 it makes up for in usability – that display is much larger than the tiny one on the original MK14. **LXF**



Housed in a 40-pin DIP package, the National Semiconductor SC/MP looked much like its rivals on the outside.

» **IMPROVE YOUR LINUX SKILLS** Subscribe now at <http://bit.ly/LinuxFormat>

# Clever CAD coding for clients and cigars

Hungarian boy genius **Tam Hanna** tidies up his underground bunker's cigar corner with a bit of creative 3D printing.



OUR  
EXPERT

**Tam Hanna's** minimal creative capability makes him ideally suited to teaching all kinds of workarounds for problems that require the use of creativity.

**T**he experiments performed in last month's tutorial proved that *OpenSCAD* can create all kinds of models consisting of primitive geometries. This is a complete understatement – in this instalment, we are modifying our approach to use 2D/3D-combinatorial techniques.

When applied correctly, they permit the creation of highly intricate models with what, in practice, could not be a more straightforward set of base geometries. While this might initially sound confusing, please take our word that nothing is quite as complex as it seems.

Incidentally, the object created in the following steps can be, and has been, used for practical purposes. When Tam needs to provide a set of cigars or candy sticks to his clients, these 3D-printed objects not only catch attention but also serve as a great first demonstration of the additive-manufacturing capabilities of his company.

Should you continue to work directly from the excursus in the last issue, click the menu option File > New File to start a new design. For reasons of simplicity, we will start afresh – our first victim, then, is the following bit of code:

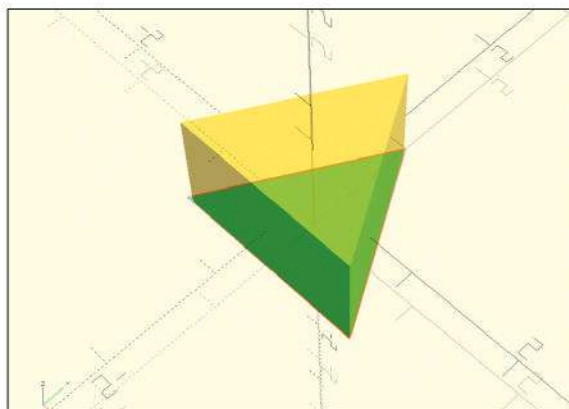
```
centeredtri();
module centeredtri(){
  polygon([[0.866,-0.5],[0.866,-0.5],[0,1]]);
}
```

The core of the method is an invocation of **polygon** with an array of three points; the mathematically inclined will quickly recognise that it creates an equilateral triangle.

Using the **module** syntax permits the geometry to be boxed up for use – the invocation **centeredtri()**; motivates the parser to create the geometry contained in the module.

Interestingly, the screenshot (*above-right*) shows that performing a preview and the production rendering now yield entirely different results.

For now, it suffices to say that methods such as **polygon** – incidentally, quite a few additional ones can be found in the documentation at [https://en.wikibooks.org/wiki/OpenSCAD\\_User\\_Manual/2D\\_Primitives](https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/2D_Primitives) – create two-dimensional objects. When a preview rendering is done, the parser assigns them some height in the direction of the Z axis to yield a more



The yellow triangle is the preview rendering; the flat dark triangle is the final rendering from the same code.

satisfying rendering. Our next job involves the creation of three instances of the triangle:

```
r = 4;
translate([-0.866*r,-0.5*r])centeredtri();
translate([0.866*r,-0.5*r])centeredtri();
translate([0,1*r])centeredtri();
```

Given that this object should become a vase, its dimensions might need to be modified depending on the intended load out. Due to this, the sequence **r = 4**; is used to introduce a variable; change needs to be done only in one place if the multiplication factor applied to the six coordinates in the three **translate** calls needs to be modified. In practice, this yields significant time savings.

## The Play-Doh press

Attempting to export a 2D-only rendering using the STL button yields the error: 'Current top level object is not a 3D object.' There must be something in the *OpenSCAD* syntax to transform the two-dimensional objects into a three-dimensional representation.

In the following steps, space constraints limit us to the **linear extrude** operation; Tam's book with the ISBN of 978-1-907920-99-8 provides a detailed overview of the various possibilities provided. For the first trial, we invoke the operation as follows:

```
linear_extrude(20)
{
```



```

r = 4;
translate([-0.866*r,-0.5*r])centeredtri();
translate([0.866*r,-0.5*r])centeredtri();
translate([0,1*r])centeredtri();
}

```

The most basic invocation requires a numerical parameter specifying the height of the extrusion. Furthermore, the chained parameter is the actual geometry to be extruded.

Applying the operation to our above-mentioned constellation of triangles, this then yields the results shown in the screenshot (right).

An extrusion operation creates multiple copies of the 2D object above one another, providing them with a three-dimensional spatial representation.

Logic dictates that these operations don't necessarily need to work on equal copies of the objects. Advanced parameters permit *OpenSCAD* to use rotational or scaling operations on each element; while a rotation obviously rotates the particular objects, scaling operations increase or decrease the size of the individual copies.

As a first experiment, we will slightly reduce the radius and apply the scale parameters:

```

linear_extrude(20, scale=3)
{
    r = 3;
    ...
}

```

In our case, setting a parameter value of 3 informs *OpenSCAD* that the final object to be placed on top must be three times as large as the one that was used at the start of the process.

In theory, you could also pass in a value smaller than one, thereby reducing the size of the objects. This can be used to create pyramid-like shapes.

After that, we will add some twist, as per the following code:

```

linear_extrude(20, scale=3, twist=180)
{
    r = 3;
}

```

**Twist** instructs *OpenSCAD* to rotate the geometry copies used in the linear extrusion process. Passing in 180 leads to a rotation of 180° – this is best seen in the screenshot showing the result (right), which – by taking a look at the object from the top – indicates how the triangles move.

### Scaling operations.

Careful readers of last month's tutorial on *OpenSCAD* may worry about the small dimensions – if the intended size is 100mm, passing in 10 does not work.

As a first test, we will thus use the benefits of parameterisation to modify the object:

```

linear_extrude(100, scale=3,
twist=180)
{
    r = 30;
}

```

Sadly, rendering this does not yield the expected result. The size of the triangles, being 1×1, produces a tiny object.

Solving this problem does not require modifications to the actual coordinates. Instead, *OpenSCAD* provides the **scale** operator, which behaves similarly to the **translate** operator used to move objects around the screen. It, however, scales the size of the target elements up and down.

Given that we want to scale uniformly along the X and Y axes, our invocation looks like this:

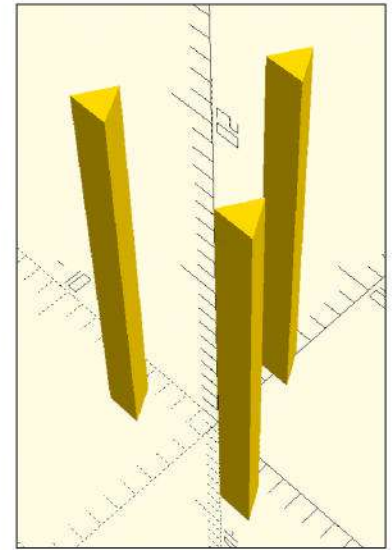
```

module centeredtri(){
    scale(10)
    polygon([[-0.866,-0.5],[0.866,-0.5],[0,1]]);
}

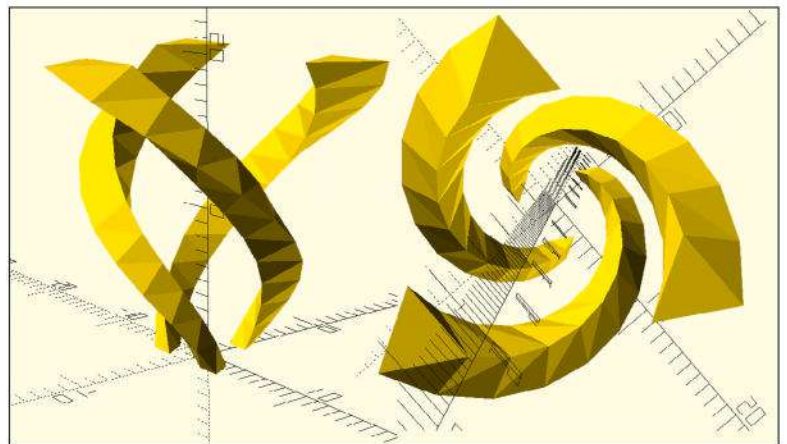
```

In practice, it is also allowed to pass in a vector describing the individual amount of scaling to be applied to the axis. Values larger than 1 obviously lead to an increase in size, while values smaller than 1 lead to a decrease.

At this point, one problem remains: a careful look at the generated object reveals it to consist of clearly visible triangles. This is a limitation not dissimilar to the



The first proto vase has been successfully erected.



Combining scale and twist leads to modern-art-like results...

## » TAM'S TOP THREE TIPS!

### Use Stack Overflow

Just in case you're wondering where the triangle coordinates came from, we didn't break out the Parker pen and scientific calculator. Instead, a quick Google search led to <https://bit.ly/lxf314centre> – this page on the Stack Overflow website provides detailed info and ready-to-use coordinates.

### Beware the Moiré effect.

A careful look at the renderings reveals weird, snake-skin-like distortions on the model. This is a common problem in computer science known as the Moiré effect, and it does not affect the model geometry when printed. More info can be found at [https://en.wikipedia.org/wiki/Moir%C3%A9\\_pattern](https://en.wikipedia.org/wiki/Moir%C3%A9_pattern).

### Go multicore!

By and large, *OpenSCAD* is an entirely single-threaded affair, so can't take full advantage of your multicore CPU. Developers have, however, been working to address this limitation for quite some time. You can find an overview of the current state of the proceedings at <https://bit.ly/lxf314hack>.



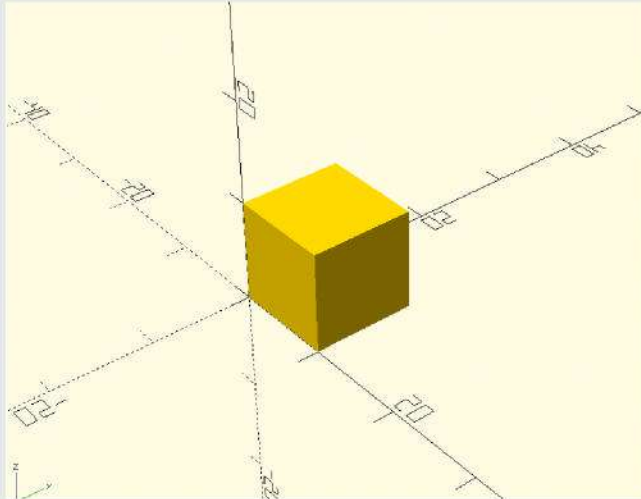
## » HARNESS THE POWER OF THE CENTER OPERATOR.

Creating the foot by subtracting a cube and some target geometry is a surprisingly common operation in the world of *OpenSCAD*. When doing such

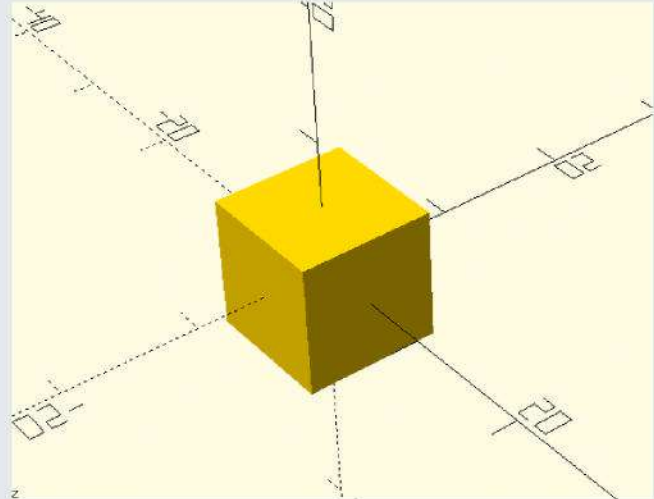
subtraction operations, use very large sizes for the cube – this way, changes in the size of the victim geometry do not directly require a modification of the subtractor.

Sadly, the default positioning of objects generated is less than ideal for cutting operations. This can easily be avoided by using the **center** object parameter,

which places the coordinate 0-0-0 in the middle of the object. To see how this works, take a look at the images below, which compare two invocations.



cube(10)



cube(10, center=true)

**\$fn** parameter discussed last month – to limit the amount of computation required to finalise the model, the number of copies placed above one another is limited. Fortunately, the **slices** parameter permits the alleviation of this situation:

```
linear_extrude(100, scale=2, twist=180, slices=1000)
{
    r = 30;
```

Selecting the correct number of slices is best done via computation; in the case of the 3D printer used to print the object, the smallest sensible layer height is 0.1mm. Dividing the intended height by the number of layers yields the parameter shown here. Incidentally, the rendering shown in the screenshot (*opposite page, top-left*) indicates that smoothness has dramatically improved. As always, creating excessive amounts of detail should be avoided – it does nothing except waste CPU time.

### A firm bottom

One of the more painful experiences of people who undergo a computer-first learning cycle involves the limitations of real-world processes. A common example would be PCB layout software, which can be configured to create printed circuit boards with geometry so small that they will remain out of commercially viable PCB production for at least a decade.

Our vase is similar in its current state – it would create three objects that would fall apart. Fortunately, this problem can be solved by printing a base.

In theory, this base could be created from a more-or-less arbitrary geometric solid. Allow us, however, to suggest a slightly different approach that showcases the power of the *OpenSCAD* description language.

The first step is creating a module called **vase** that will contain the – so far freestanding – markup for creating the vase structure. After that, the following bit of code is used:

```
scale(1.1)hull()foot();
vase();
module foot()
{
    difference()
    {
        vase();
        translate([0,0,101])cube([200,200,200], center=true);
    }
}
module vase()
{
    ...
}
```

While the markup is quite long, it is actually very simple. Firstly, the module **foot** uses an enormous cube to cut out a small slice of the entire vase structure.

In the next step, the results are fed into the **hull** operator – **hull** is a volumetric operator that wraps the geometry in its entity.

Finally, the resulting object is scaled to create the structure shown (*opposite page, top-right*).

In the next step, a similar approach must be taken for the lid. In that case, however, the **hull** operator is slightly critical – it would create a solid plane. Fortunately, this problem can be solved by creating arbitrary geometry for the free space. This leads to the following structure:

```
scale(1.1)hull()foot();
vase();
difference()
```



Things look much smoother after the adjustment.

```
{
hull()roof();
linear_extrude(2300,center=true)rotate(180)scale(4.5)
centeredtri();
}
```

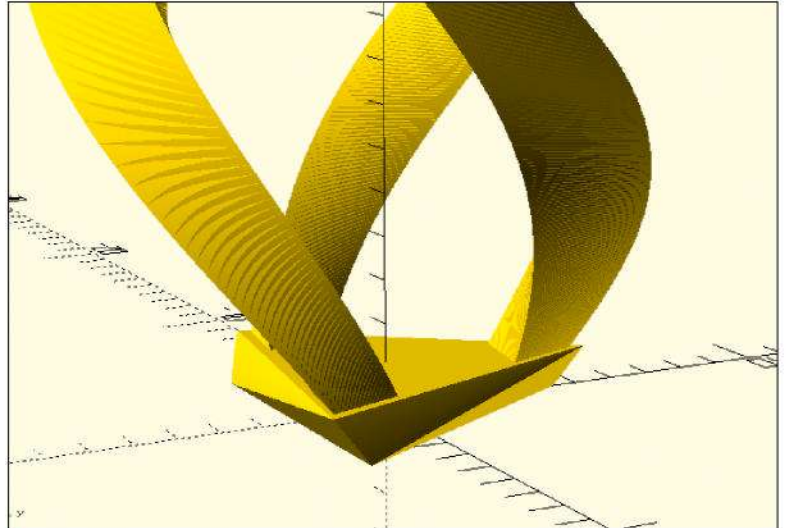
## Generating text

While 3D-printed objects tend to speak on their own, versioning them can be helpful. The example of a 3D-printed tag bearing the name of the person owning the keys is heavily overused, but adding a little engraving to our holder might be interesting.

Given that the rendering of these (relatively complex) objects takes quite a bit of time due to *OpenSCAD* not being parallelised, we need to start out by opening another tab. In that, we can start playing with the text-generator functions. Try the following:

```
text("LXF");
```

*OpenSCAD* provides dozens of parameters, documented in detail at [https://en.wikibooks.org/wiki/OpenSCAD\\_User\\_Manual/Text](https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Text). For us, however, only the **size** parameter is relevant – by default, it is 10,



This stand will improve the robustness of the design.

which leads to a 10-*OpenSCAD*-unit-high basic appearance of the generated text.

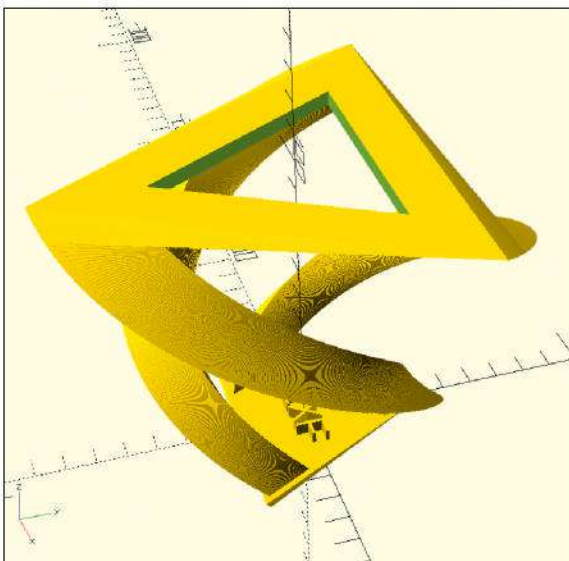
Sadly, the rendering engine does not provide a way to set the font width. Positioning thus has to be done by trial and error – for our vase object, the following modification led to a working model:

```
translate([-10,-5,0])linear_extrude(5)text("LXF");
scale(1.1)hull()foot();
vase();
```

At this point, it is time for the final production rendering. The final screenshot (*below-left*) shows that our object is ready for print.

Even though two issues of *Linux Format* are not nearly enough to do justice to *OpenSCAD*, our guides have demonstrated its basic operational handling.

We hope that the experiments shown here motivate you to analyse the software further. As in the case of many other pieces of open source software, *OpenSCAD* is one of these technologies that one either loves or hates. The experiments performed here are enough to determine on which side of the camp you will reside. **LXF**



The object is ready for deployment, alongside the real-world finished item.

» **IMPROVE YOUR LINUX SKILLS** Subscribe now at <http://bit.ly/LinuxFormat>



## PROTON-GE

Credit: <https://github.com/GloriousEggroll/>

# How to fix glitching Steam Deck games

Despite amazing work, there's plenty of incompatible Steam Deck games, thankfully fixes are at hand, but nothing so far for **Neil Mohr**.



**OUR  
EXPERT**

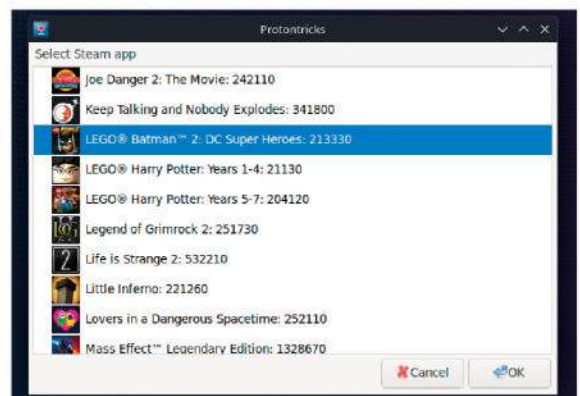
**Neil Mohr**  
still can't pry  
his Steam Deck  
from his six-year  
old's hands.

**T**here's no doubt Valve Software has done more to propel Linux gaming forward than almost anything else in recent years. The Steam Deck and its *Proton* spin of *Wine* have brought more games to Linux along with new users than anything else could. But despite the stellar work put into fixing game compatibility, glitches still exist.

Some simply won't ever be fixed, but for many, the Steam community has chugged into action and created a host of workarounds. We're looking at a number of troubleshooting options here, from spotting them before you even try a game to replacements for the default *Proton* layer.

Let's kick off with basic troubleshooting, so we know about issues before even buying a game, never mind running it and trying to play it. You should head to [www.protondb.com](http://www.protondb.com) and search for the game in question. We're going to be looking at *Lego Batman 2*; this is rated as Unsupported by Valve for the Steam Deck, while ProtonDB has it classified as Silver. Why the two different classifications?

ProtonDB uses the original *Wine* classification for software compatibility. Basically, if it runs, it gets a Bronze rating, even if there are serious flaws and crashes. A Garbage rating implies it's unusable or won't run at all. Silver means it runs well but there's the odd major glitch and it requires workarounds. Platinum is a



As long as you add access to your SD card, you'll get a full list of supported games to tinker with.

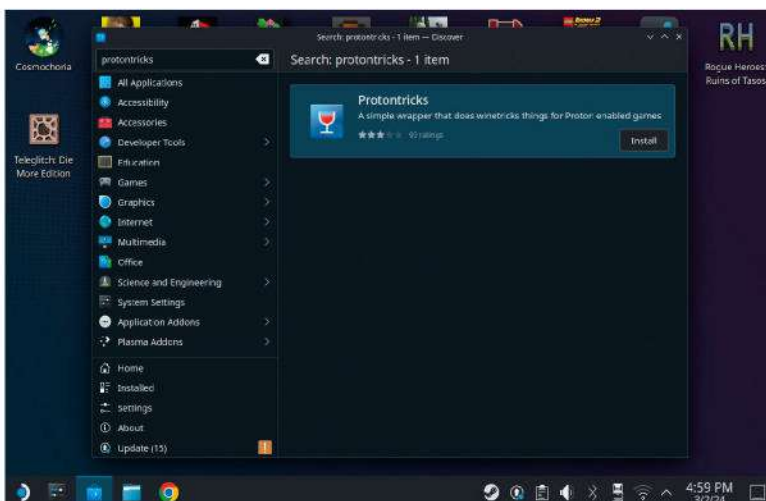
flawless experience, while Gold is a step down from that, with perhaps just a minor bug requiring workarounds. Valve takes a more formal approach – a game labelled Unsupported means “Your game does not function on Deck due to incompatibility with *Proton* or specific hardware components.” Basically, Valve tested it, found issues and slapped Unsupported on it as it doesn't plan to fix the issues in *Proton* at this time. It doesn't mean the game won't run.

The best way to highlight this is with a look at a real game. If you try to run *Lego Batman 2* – classified as Unsupported on the Steam Deck with even the latest *Proton 9.0* or *Proton Experimental* – you'll find loads of textures are daubed with a bright rainbow flickering effect. The game works but it's visually unpleasant.

On the ProtonDB website, if you click on the *Lego Batman 2* entry, you'll find a list of comments from users reporting play issues and potential fixes. The first fix is to use *Protontricks*. This is a software wrapper for *Winetricks* that provides a bunch of standard fixes – such as listing missing or alternative DLLs – for Windows games.

Let's install *Protontricks* and go through the process of fixing a game. To get it installed, ensure you're out of Family View (the green family icon is displayed at the top if you are), press the Steam button > Power Off > Switch To Desktop Mode. Remember to open the on-screen keyboard (press Steam+X) – the right

In Desktop mode, Steam Deck uses Discover to install Protontricks and get this ball rolling.





touchpad is your mouse. Select Menu > System > Discover and type **Protontricks** into the search box.

Select the **Protontricks** entry and click Install. Once done, select Launch. Helpfully, it scans your Steam Deck drives and presents a list of known games with solutions. If you have an expansion drive – an SD card – it needs permissions to scan this, so copy and paste the presented text into the terminal and rerun **Protontricks**. It'll look similar to this:

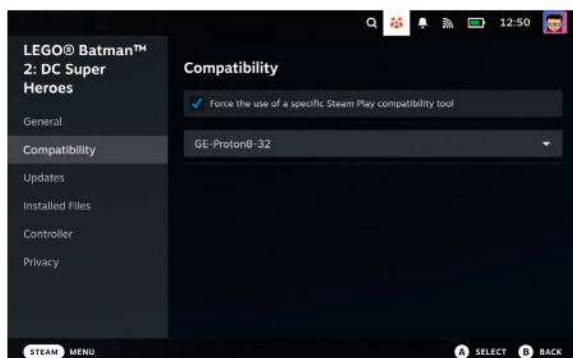
```
$ flatpak override --user --filesystem=/mnt/MySSD/SteamLibrary com.github.Matoking.protontricks
```

**Protontricks** works from within Desktop mode, so when it comes to fixing our game, click Start > Utilities > **Protontricks**. It presents a list of recognised games, so find the one you want to fix, select it and click OK.

As this is a **Winetricks** wrapper, you're taken to the **Winetricks** interface, which isn't the most user-friendly, but let's see how it works. Select Default Wineprefix > Install A Windows DLL and choose the DLLs or required components. When you click OK, nothing may happen for a while – just wait and OK any dialogs. If you're installing Windows components (such as Visual C++), go through the install process. In our case, we selected d3dx9 and d3compiler\_47. Once back to the **Winetricks** interface, you can close it, as it's done the job of adding these new files to the game directory. Be aware you can't remove DLLs, you have to remove the entire game. You can try by running **Protontricks**, select Default Wine prefix > Change settings > select the box allDlls=default then click OK, but it's unlikely to work. This is a Windows issue rather than a Proton issue.

## Proton GE

We've looked at **Protontricks** first as this is a user-configurable option that can be applied to any Steam (and non-Steam) games installed, but it's a bit of a faff



Ensure you engage the Proton GE layer to fix your gaming issues.

to use. Enter **Proton GE**. This is a community build of **Proton** that usually has all of the **Proton/Winetricks** fixes in place. This means once it's installed, just select the alternative **Proton GE** layer and you're good to go from the standard Steam Deck interface, with no Desktop messing around. Well, just for the install part.

As before, enter Desktop mode – hold the Steam Deck's Power button and select the Switch To Desktop option to open up the Linux desktop. We need to install **ProtonUp-Qt**, so select Menu > System > Discover, and type **ProtonUp-Qt**. Tap or click Install, and wait for it to finish installing.

Once it's installed, tap on the store page, then hit the Launch button in the top-left corner. Add **Proton GE** – in the **ProtonUP-Qt** interface, tap the Add Version button. The first drop-down box should read "Proton-GE", while in the second (Version), you want to select the latest version available, then click the middle Install button. If it's been installed correctly, the Installed list of **ProtonUp-Qt** now has a new entry for **Proton GE**. There is no auto-update option, though, so check back here regularly for **Proton GE** updates.

At this point, you can log out of the Desktop back to the main Steam Deck interface. To force a game to use **Proton GE**, select the game to open its Launch screen and click the cog icon, then select Properties > Compatibility. Click Force The Use Of A Specific Steam Play Compatibility Tool. Open the drop-down menu below and at the top will be **Proton GE** with the version number you chose to install. Press B to go back and play your now perfectly running game! **LXF**

The **LEGO Batman 2** game before and after glitches have been fixed.

## QUICK TIP

If you need details of game components, head to [www.steamdb.info](http://www.steamdb.info), search for your game, select it, click the Depots link on the left and then the ID number. This lists all the associated support files.

## »» DESKTOP STEAM

While we might be writing about the Steam Deck here, the same tools work on standard desktop Linux distros. As desktop Steam still uses the **Proton** layer, you can choose different compatibility layers in the same way through the game's Properties interface. You can also use **Protontricks** to add and remove Windows components to fix issues with specific games. The official **Protontricks** Git is at <https://github.com/Matoking/protontricks> but you might find this streamlined guide that only uses **Pipx** easier to follow: <https://simpler-website.pages.dev/html/2021/1/protontricks>.

As for **Proton GE** on desktop, there is a version of **ProtonUp-Qt** available as both an AppImage and a Flatpak. Go <https://davidotek.github.io/protonup-qt/> to get details on how to install both options.

»» **GET LXF IN A PORTABLE VERSION** Subscribe now at <http://bit.ly/LinuxFormat>

## Tips for managing Docker containers

Generous-minded **Stuart Burns** has years of Docker experience under his belt and is keen to share his knowledge with other sysadmins.

**E**veryone knows how containers revolutionised application building and deployment. Using a disposable stack of containers that make up an app that aren't using the `docker-compose` command to manage the stack are missing a trick.

It allows the shipping of an entire stack in one single file. Using `docker-compose` enables the administrator to group all the application's containers into a single stack. It means an entire stack can be started or torn down with next to no effort. No configuration needed.

It can be as simple as `docker-compose up` and stopping the stack with `docker-compose down`. As a really useful example, it provides a way to distribute an entire application that can be deployed as designed, tested and then brought down and deleted.

In production, using `docker-compose` is much easier than trying to launch 15 different containers manually and the overhead that comes with that. Many applications now provide the `docker-compose.yml` file to build the application rather than raw *Docker*.

Using this method also makes networking easier, because it places all the containers in the same network, making communication simple and less problematic. Below is a really simple setup to show how useful and easy it is to utilise.

Depending on your setup, you may need to install the *Docker-compose* application. In the case of Ubuntu, this would be:

```
$ sudo apt install docker-compose -y
```

The *Docker-compose* file is a YAML file that holds the configuration required. It details how to set up the containers. A trivial one-container-file YAML file could look like this:

```
version: '3'
services:
```

```
apache:
  image: httpd:2.4
  ports:
    - "8080:80"
  volumes:
    - ./html:/usr/local/apache2/htdocs
```

To run this base image of Apache, copy the YAML file and put it in a file called `compose.yml`. Also, create a folder called `html` and create a rudimentary `index.html` in the folder with a bit of HTML markup.

From within the folder, use the command `docker-compose up -d` and it will use that information to create the entire stack. The `-d` parameter tells the system to run it in the background. To bring it all down, use `docker-compose down`.

As you can see, most of the items are self-explanatory, with sections for networking and volumes. You can modify to your own tastes.

For casual users, it makes the whole management easier. Rather than trying to remember all the switches, trawling the history or creating scripts to start the apps, it is as easy as `docker-compose up -d` from inside the folder. Below is what a multi-container example might look like:

```
version: '3'
services:
  webapp:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html

  database:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: examplepassword
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: myuser
      MYSQL_PASSWORD: myuserpassword
    ports:
      - "3306:3306"
    volumes:
      - ./data:/var/lib/mysql
```

Recently, we had the rather dull task of creating a *Docker-compose* file for a stack that contained several

Being able to launch a series of complex deployments with a single command is admin heaven.

```
sysadmin@dockhost:~/dockhost/daily$ docker-compose up -d
Creating network "daily_default" with the default driver
Creating gitea ... done
Creating portainer ... done
Creating webserver ... done
Creating dozzle1 ... done
Creating dashy ... done
Creating jellyfin1 ... done
sysadmin@dockhost:~/dockhost/daily$
```



Docker instances to bring it up. Previously, there was a series of `docker run` commands. However, we can do better!

We could have manually done all the work of creating a new YAML compose file from scratch. Some of these `docker run` commands took up four lines.

We didn't. Using our internal ChatGPT AI system, we asked it, as an expert in Docker and containerisation, to please take the following Docker commands and create a Docker-compose YAML file instead. It needed a bit of tweaking, but apart from that, it worked without issue. Work smarter, not harder.

Pro tip: refactoring long lines of code is quite easy with AI systems.

While fixing all this, we were asked, "Why do the containers never restart on boot even though they were set to 'restart unless stopped'?"

The answer is in the restart policy and the definition of 'restart unless stopped'. The top and bottom of it is, when a system is rebooted, the containers are stopped before shutdown, satisfying the stopped condition. If you want a container to start on boot, assuming the Docker service is started, it is as simple as using the restart policy of always.

It would obviously be remiss not to mention that including the restart policy for each container is as simple as adding a line in for each, an example being:

```
restart: always
```

One last note: YAML doesn't like tabs. Don't use tabs. Using them will stop Docker-compose from reading it properly. One really useful tip: we use <https://codebeautify.org/yaml-validator> to sanity check the layout of a YAML file before using it. Obviously, don't leave in sensitive information, but it's a really useful site. **LXF**



Docker containers have transformed the way everyone handles deployments and updates.

```
GNU nano 6.2
--
version: "2.1"
services:

  apache2:
    container_name: webserver
    image: httpd:2.4
    environment:
      - TZ=Etc/UTC
    volumes:
      - /encrypted/serv:/usr/local/apache2/htdocs/
    ports:
      - 8084:80

  portainer:
    container_name: portainer
    image: portainer/portainer-ce
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /encrypted/portainer/portainer_data:/data
    ports:
      - 9443:9443
      - 8000:8000

  dozzle:
    container_name: dozzle1
    image: amir20/dozzle:latest
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
    ports:
```

Behind-the-scenes configuration is easy via JSON files.



**Stuart Burns** is a Linux administrator for a Fortune 500 company specialising in Linux.

## » DEBIAN VERSUS UBUNTU

Last month, I was rather vocal in my criticism of Canonical. Following that, I decided to go back to a 'pure' experience. I decided to run Debian 12 on my main PC. It was very different.

I respect what the Debian team does and excluding non-free binary blobs. That said, the experience just wasn't as polished as Ubuntu or other popular distros. It took work to achieve a similar setting and I was one of the users who got caught with the recent Nvidia binary blob, so no local AI generation for me. For several days, there was no real fix.

It shows how complex the software is under the surface. Binary drivers just make life easier. Experiencing these issues and with a deadline, I decided to revert to Ubuntu. Everything just worked.

While we may dislike binary blobs and the fact that big business is involved in our open source, it makes life easier. But without Debian, we wouldn't have Ubuntu and the IT world would be much duller.

Debian is great as a stable distro that works and is reliable, with super-long support cycles. But, as they say, "Debian ain't Ubuntu." Users trying to install packages designed for Ubuntu package managers tend to break stuff. So much so that Debian has a page called Don'tBreakDebian: <https://wiki.debian.org/DontBreakDebian>

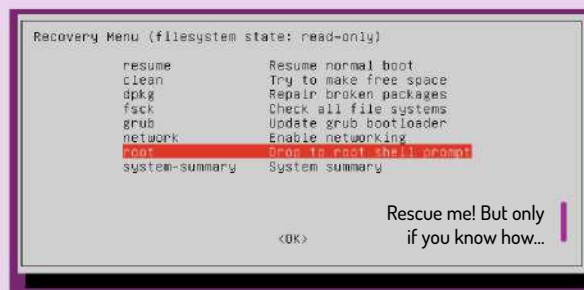
Finally, I'd forgotten how difficult it can be to purchase a guaranteed-to-work Linux desktop. This resulted in me buying a new desktop from my local provider and spending an extra £90 on a mobo with an Intel network and wireless, making a decent experience.

## » ROOT PASSWORD RESETS SIMPLIFIED

Use the GRUB menu during boot! There, we're done. Oh, keep reading...

We're sure we're not the only ones who do an install and need to reset the root password. It can be a chore having to boot up, remount the filesystem, change the password etc. Some vendors even include such issues in their exams, and those who are unprepared lose precious time.

For a while now, a lot of vendors, like Ubuntu, have a boot menu that allows you to manage these things. When booting up, hold the Shift key (for BIOS



systems) or the Esc key (for UEFI systems). It presents the GRUB menu. Choose Advanced Options and then one of the Recovery Mode options – usually for the latest kernel version. This enables the low-level rescue mode,

including resetting the password, booting to a previous kernel (for example, if you applied a kernel that has issues that prevent boot), and more.

It's a very useful thing to know about, but it seems a lot of people are unaware.

# Collaborate with OnlyOffice DocSpace

Nowadays, online document collaboration is a must. OnlyOffice DocSpace allows organising this effectively. **Kseniya Fedoruk** shows how to set it up.

**QUICK TIP**

To enable the AI assistant, access the Plugin Manager in the opened editor via the Plugins tab, find ChatGPT and click Install.

Every day, we work with numerous office files together with our colleagues, team members and various external users. Sometimes (or maybe often) chaos breaks out; the necessary editing rights are missing, important files can't be found, or co-workers from another department don't leave suggestions but edit the text directly. Such a scenario can cause serious problems, especially when it comes to sensitive documents such as contracts.

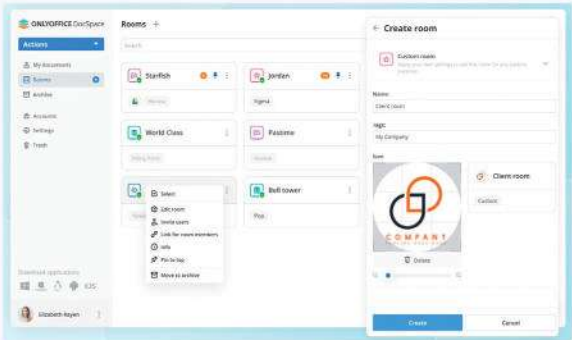
Keeping this in mind, it's important to choose a solution that helps eliminate such difficulties. In this guide, we'll introduce you to the features of OnlyOffice DocSpace and explain how to install the server build on your Linux machine.

OnlyOffice DocSpace is a room-based environment for organising a clear file structure depending on your needs. The platform comes with integrated online viewers and editors, enabling you to work with files in multiple formats, including text docs, digital forms, sheets, presentations, PDFs, ebooks and multimedia.

### Your new room

DocSpace rooms are group spaces with preset access levels to ensure quick file sharing and avoid unnecessary repeated actions. Currently, among the provided room types there are Collaboration, Public and Custom rooms.

Collaboration rooms are intended for co-authoring documents, tracking changes and communicating in



Creating a Custom room in OnlyOffice DocSpace.

real time. In Public rooms, external users can view documents without registration. Custom rooms are designed for any custom purpose – for example, to request document review or comments, or share any content for viewing only.

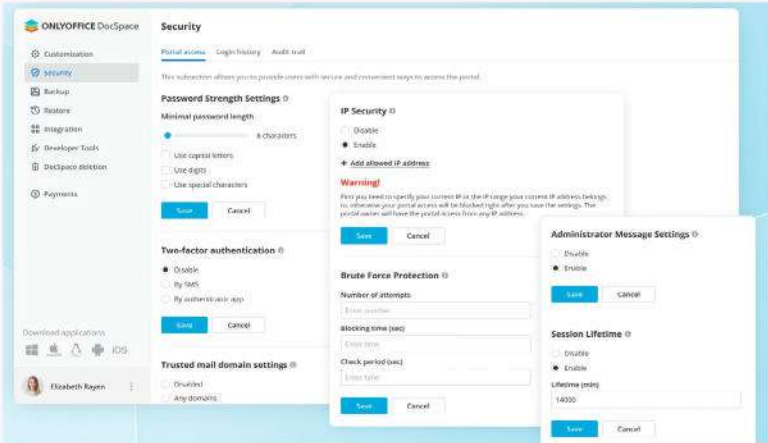
Flexible access permissions enable you to fine-tune the access to the whole space or separate rooms. So, available actions in a room depend on the given role.

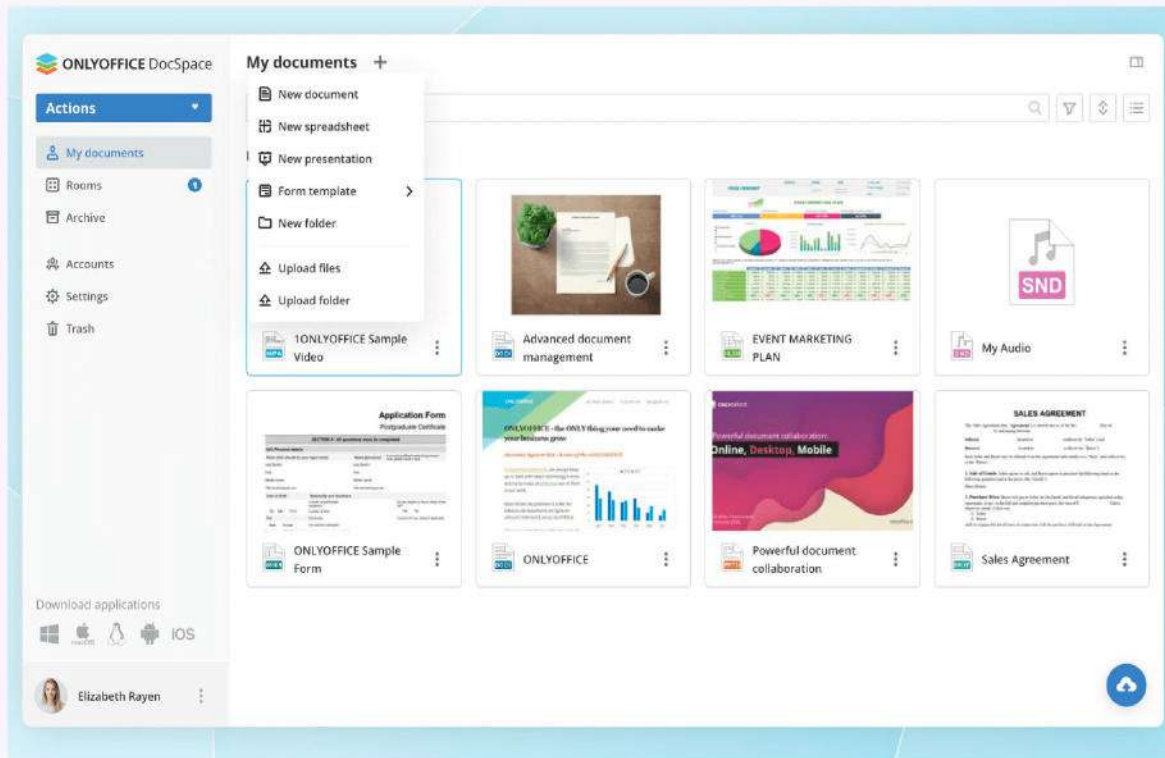
Role	Function
Owner	Creates and manages DocSpace
DocSpace admin	Manages DocSpace
Room admin	Manages rooms
Power user	Can create and store their own files
User	Only has access to the rooms they are invited to; can't create or store files
Editor	Can edit docs
Viewer	Can view docs
Reviewer	Can suggest changes
Commenter	Can leave comments
Form filler	Can fill out forms

### AI helper

OnlyOffice DocSpace comes with an AI helper in the form of a ChatGPT plugin accessible right in the

Security settings in OnlyOffice DocSpace.





Working with various content types in OnlyOffice DocSpace.

### QUICK TIP

Developers and integrators can make use of the open API of OnlyOffice DocSpace to integrate the whole space or separate rooms into their own web applications: <http://api.onlyoffice.com>.

editors. It allows running word analysis, generating images and keywords, summarising, translating and rewriting texts, fixing grammar and spelling, finding synonyms, building code and so on.

Customisation settings enable you to tailor OnlyOffice DocSpace to your liking. You can set the language and time zone, change the space name and URL, add a description for the Welcome page, upload your own logo and select the colour scheme.

## Security

OnlyOffice DocSpace is a secure way to work with both personal and company documents. HTTPS protocol encrypts traffic in transit, JWT protects files from unauthorised access, and it's also possible to enable password protection and watermarking.

Along with various access levels, space admins are also able to adjust the security settings to control the login procedure and prevent data leaks, including such options as 2FA and single sign-on, trusted mail domains and session lifetime, IP restriction and so on. To monitor user activities, Login History and Audit Trail features are available. Automatic and manual data backups ensure your data remains safe in case of any infrastructure failures.

## Installing DocSpace

Installation of OnlyOffice DocSpace on Linux servers is available using an automatic script with Docker installed. minimum system and OS requirements are:

- **Quad-core CPU** (six-core recommended)
- **8GB RAM** (12GB recommended)
- **40GB** of free HDD storage space
- **6GB or more** of swap space
- **64-bit (AMD64)** Linux distro with kernel 3.10 or later

Download the Docker script file with:

```
$ wget http://download.onlyoffice.com/docspace/docspace-install.sh
```

Once ready, make sure you have the root rights and continue. You can install OnlyOffice DocSpace from the DEB or RPM package with:

```
$ bash docspace-install.sh package
```

Or by using the Docker script with:

```
$ bash docspace-install.sh docker
```

Once finished, launch your browser and type the local network computer IP address into the address bar. On the DocSpace setup page, enter your email address, set a password, and accept the licence agreement. At this stage, it's also possible to set the language and time zone, choosing the necessary option in the corresponding drop-down list. When all the parameters are set, click the Continue button.

That's it! OnlyOffice DocSpace is perfect for organising effective and protected collaboration with all types of users. Install it on your own Linux server to get complete control over your data and its location. **LXF**

## » SELF HOSTING

OnlyOffice DocSpace can be hosted on your own server as well as in the trusted public cloud. For the cloud version, there is a completely free Startup plan for individual users and smaller teams. If you would like to opt for a Business subscription, you only need to pay for admins and power users with extended rights. All other regular users can be added for free. The self-hosted version provides a free DocSpace Community. There is also a Family Pack for home use, as well as DocSpace Enterprise for bigger teams and companies. To create your own OnlyOffice DocSpace, visit the [www.onlyoffice.com](http://www.onlyoffice.com).

» **GET YOUR MONTHLY ADMIN FIX** Subscribe now at <http://bit.ly/LinuxFormat>





## CHATGPT MASTERCLASS

# Make AI work for you

Want to get more from the best-known AI service on the planet? **Barry Collins** and **Neil Mohr** explain how to make AI work for you.

**O**penAI's now infamous chatbot has a series of formidable features, most of them hidden from those who don't subscribe to *ChatGPT Plus*. It's worth doing, even if only for a month, so you can evaluate the sheer power that lurks here and work out whether it could make a difference to your professional – or even your social – life.

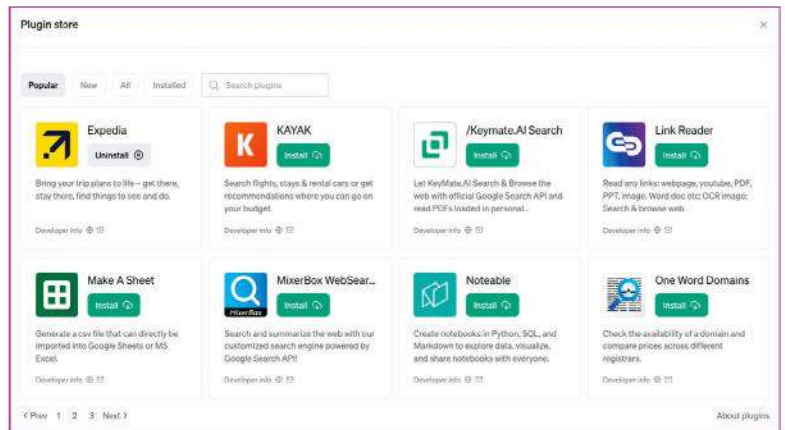
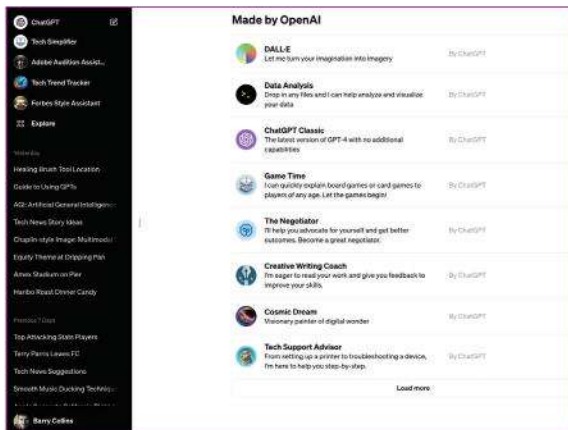
*ChatGPT Plus* can perform hugely sophisticated data analysis, saving you from having to spend time learning how to do it yourself in applications such as *Excel*. It can tidy up your computer code, your emails or your *PowerPoint* presentations. It can hook into third-

party services to plan holidays, book restaurants or scour for facts on your behalf.

But unless you know how to get the best out of basic AI prompts you aren't going anywhere fast. We'll explore prompt engineering, the fascinating area of getting the best out of AI's by formatting, one-shotting and chaining prompts to manipulate the AI into generating evermore complex responses.

Then there are GPTs – mini chatbots. Create and train your own AI without having to bash out a single line of code. Made with plain English commands they can be trained on your own data, so are highly personalised to you or your business.





There can't be a reader out there who hasn't spent at least 10 minutes noodling with *ChatGPT* – or one of its many derivatives, such as the *Bing Copilot*.

While there's plenty of AI power and much to explore with the free version of *ChatGPT*, the real power is unlocked when you subscribe to *ChatGPT Plus*. Not only does that (largely) ensure you can get access to the chatbot during the regular periods of peak demand when the free version is switched off, it opens up extra features such as plugins and advanced data analysis. Let's look into those two features in greater detail.

## Extending services

Plugins enable you to combine the intelligence of *ChatGPT* with third-party sites or services, such as Wikipedia, WolframAlpha (this fixes *ChatGPT*'s poor mathematical issues) or OpenTable. For example, you could ask the AI to create a summary of a long article you've found on Wikipedia or use it to research facts. For instance, you could ask ChatGPT to "give me the release dates of the past five versions of Windows from Wikipedia" and it will dig out the correct answers from the online encyclopedia, without you having to wade through different articles yourself.

With the Expedia plugin, on the other hand, you can ask the AI to do things such as create a three-day trip to Oslo for you, and it will come back with hotel and flight recommendations that suit your budget. It's not perfect – it sometimes gets trip dates wrong when you click through to book flights, for instance, and it only

deals with the US site – but it's good for generating ideas and rough budgets.

Irritatingly, you can only enable three plugins at a time and the Plugin Store is poorly presented and erratic. Often plugins don't install at the first time of asking, for example. It's a work in progress, but one that holds tremendous promise if it can be made to function more smoothly.

## Advanced data analysis

This is arguably the best feature of *ChatGPT Plus*, but you may still need to dive into Settings > Beta Features and switch it on first. Advanced data analysis lets you upload spreadsheets and other files, and get the AI to do the hard graft that you might normally do in Excel.

For example, we fed *ChatGPT Plus* raw survey data into an Excel workbook. Each tab represented a different category, and the AI automatically figured this out, but checked this with us to ensure it was making the correct assumption.

Once it worked out the format, we could ask it to produce graphs showing each company's performance in different categories or to tell us how company A fared against company B in different categories. We could also use it to calculate different weightings for each category, for example giving 'very satisfied' responses a +2 weighting, but mere 'satisfied' a +1, to see what effect that had on the overall results.

All this could be done in Excel, but only if you have the skills with pivot tables or can apply the relevant formulae. Advanced data analysis lets you do all this

Above: Extend ChatGPT's abilities with specific data sets via its Plugins store.

Above-left: Commercialisation of ChatGPT is ongoing through mini GPTs.

## » WHAT ARE GPTS?

*ChatGPT* is based on the GPT-4 large language model (LLM), which OpenAI released in 2023. And a GPT is effectively a mini version of *ChatGPT*, dedicated to a specific task, such as helping you plan a gap year or how to use a piece of software.

You don't need any coding skills to create a GPT. They can be created using plain English commands given to *ChatGPT*. That doesn't mean there aren't particular skills and techniques you can apply to getting the best out of

them or training them better, as we reveal in our guide.

The key with GPTs is thinking of a very specific task that could benefit from AI assistance. It might be helping you to formulate your company's annual report or buffing your Python 3.12 skills, but the more specific the better.

You can upload files to GPTs to help with their training. Let's say you've treated yourself to a Canon EOS R50 camera and want to learn how to better use its features. You can upload the PDF

of the camera's official manual to a GPT and then use your AI assistant to guide you when you want to find out how to change the flash settings or activate the self-timer. That means the AI is less reliant on the unreliable information it might otherwise source online.

At a more advanced level, you can connect a GPT to a third-party service such as Zapier and use it to query other services, such as your Google Calendar. Zapier has advice on how to set this up at <http://zapier.com/blog/gpt-assistant>.





results from AI tools is a skill – one that can pay handsomely, too.

We'll take a quick look at how you can best format your questions to *ChatGPT* to ensure you get an expert reply in a format you want. Once this is under your belt, we'll then look at ways of asking *ChatGPT* to revise what it's initially generated to produce revised versions better suited to what your needs are. Keep in mind this revising can push AI's down the path of hallucinations (false information). *ChatGPT* and similar are just (albeit incredibly complex) statistical models, so you can push it down a path where its output is factual gibberish although still understandable English. Hence it outputting non-real cases for lawyers and non-existent Wikipedia articles complete with made-up URL links.

One of the most powerful and straightforward engineering tricks is to form your prompts in this style:

**Acting as a [ROLE] perform [TASK] in [FORMAT]**

Hopefully, you can see where this is going, but this prompt does a fantastic job of guiding *ChatGPT* down a suitable path (the role) and producing a useful response (the task) in an output (the format) that's useful for you. You can focus the output somewhat at this initial stage by expanding this basic prompt:

**Acting as a [ROLE]. Perform [TASK with OBJECTIVE] to [LENGTH]. Include relevant [DATA and SOURCE] using the style [VOICE] aimed at [AUDIENCE].**

Example roles you can input are: marketer, copywriter, accountant, lawyer, analyst, doctor, project manager, therapist, journalist, financial officer or prompt engineer. This grounds the output in an area of speciality. If you think of *ChatGPT* as a statistical

Sophisticated visualisations are possible – you just have to ask.

with plain English commands, and the results are much more reliable than they are with general chats. There are no hallucinations (at least none we've seen) and it will tell you if it can't perform a particular request or needs more guidance on what you're looking for.

## ChatGPT prompt engineering 101

You'd never expect to just rock up to a new programming language and pick up how to get the best from it without knowing the syntax, right? AI prompt engineering is just that, understanding the syntax and semantics of what the AI neural network expects and can process as input variables and commands. This prompt engineering is often sneered at in tech circles, but knowing how to get the best

ChatGPT can tackle complex topics by discussing issues with itself.

Step	Expert 1's Thought Process	Expert 2's Thought Process	Expert 3's Thought Process
1	<b>Identify Key Causes:</b> Recognize that key causes of global warming include carbon emissions, deforestation, and industrial pollution.	<b>Acknowledge Key Causes:</b> Agree with Expert 1's identification of key causes and emphasize the importance of addressing them comprehensively.	<b>Assess Key Causes:</b> Evaluate the significance of each cause, noting that some may have more immediate impacts than others.
2	<b>Address Carbon Emissions:</b> Propose transitioning to renewable energy sources, promoting energy efficiency, and implementing carbon pricing mechanisms.	<b>Focus on Deforestation:</b> Suggest prioritizing reforestation efforts, protecting existing forests, and promoting sustainable land use practices.	<b>Tackle Industrial Pollution:</b> Highlight the need for stricter regulations on industrial emissions, promoting cleaner production processes, and investing in green technologies.
3	<b>Evaluate Solutions:</b> Discuss the feasibility and effectiveness of renewable energy adoption, considering factors such as infrastructure requirements and economic incentives.	<b>Consider Ecological Impact:</b> Explore the ecological benefits of reforestation in mitigating carbon dioxide levels and preserving biodiversity.	<b>Assess Economic Viability:</b> Analyze the economic implications of transitioning to cleaner industrial practices, considering potential costs and benefits for businesses and consumers.

Message ChatGPT...

ChatGPT can make mistakes. Consider checking important information.



model, it's giving it a better chance of outputting the response you're after. Obviously, with areas involving legal consequences and medical issues, never rely on an AI and always seek professional human advice.

Keep in mind that *ChatGPT* is trained on internet-based information – subjects from around the globe. So, if you're trying to elicit local-specific advice (for example, on legal or accounting matters), ensure you specify where you are but also keep in mind its output may not apply to your situation or could be outdated.

The task element is the more complex part, as it's whatever problem you want solving, answering or creating. You don't have to go into full detail at this point; as we'll see, we can expand and refine *ChatGPT*'s output further as it remembers your responses (4K tokens in *ChatGPT 3*, 8K *ChatGPT 4*, 32K for *Pro*), but for basic tasks, you could use options such as: write a headline, create a presentation, write an essay, book outline, email, create social media post, production description, write a cover letter, blog post, summarise this, video script, create advertising copy and so on.

Finally, *ChatGPT* is able to format its output in many ways – both standard text document formatting as well as spreadsheet options, programming code, file types such as PDF and markup formats including HTML and Markdown. Often, *ChatGPT* can autoformat output, but you can try: table (with rows and columns specified), list, summary, HTML, spreadsheet, CSV, plain text, rich text, PDF and so on.

## Prompt chaining

Once you've prompted *ChatGPT*, it often outputs a ream of text that may or may not be what you're after. You can start refining its responses and asking it to generate additional output that's based on the original topic and role. For example, we could try:

act as a magazine editor create the contents for a 100 page Linux Kernel magazine in plain text

The output is unusable twaddle, with *ChatGPT* trying to launch *Linux Kernel Chronicle*, *The Kernel Gazette* or something similar. But let's get it to polish up this, erm, terrible tome.

write ten engaging names for a Linux kernel magazine  
take the best name and write a list of headlines for a blog launching the magazine  
write a list of subheadings  
write a list of call-to-action  
combine the best headline, subheadings and call-to-action write a blog launching the magazine  
rewrite the blog in 200 words

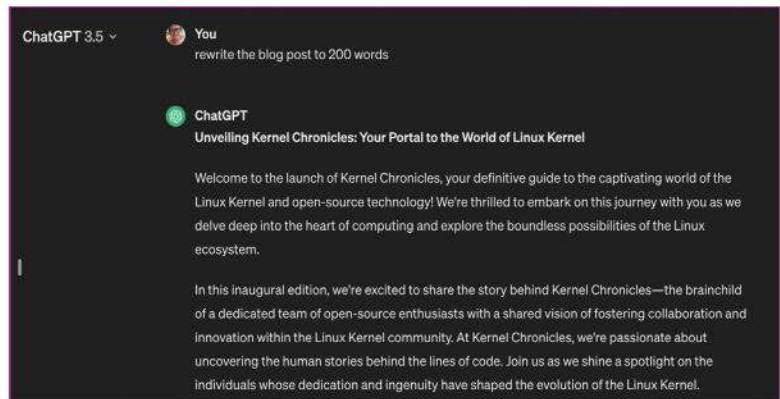
We expect in this process that you'd probably want to be more specific about what you think is "the best" rather than entirely rely on *ChatGPT* to decide everything, but it's a fun process. The final rewrite prompt can be very powerful when used in the form:

rewrite [YOUR TASK] in the [STYLE], [TONE], [VOICE] and [PERSONALITY].

You can also ask it to rewrite in first, second and third person, along with specifying word counts.

## Tree of thought

Proposed in a paper March 2023 (<https://arxiv.org/abs/2305.10601>), the tree of thought is a self-referencing problem-solving mode for LLMs. It builds on the chain of thought process and effectively gets



*ChatGPT* to brainstorm with itself. A framework for this looks like the following:

Three experts with exceptional logical thinking skills are collaboratively answering a question using a tree of thought method. Each expert will share their thought process in detail taking into account the previous thoughts of others and admitting any errors. They will iteratively refine and expand upon each other's ideas, giving credit where it's due.

The process continues until a conclusive answer is reached. Use step by step thinking and organise the entire response in detail as a table.

Provide a summary of the recommendations. My question is how can we fix global warming.

The output is a rich thought-through plan of action with a train of thought and arguments you can follow. You can also interrogate *ChatGPT* further on specifics. Hopefully, you can now see what's achievable. **LXF**

Asking *ChatGPT* to rewrite or summarise its own output is rewarding.

## » CHAIN OF THOUGHT

Chain-of-thought (CoT) prompting breaks down problems into intermediate steps, enabling LLMs to tackle complex reasoning that can't be solved with zero-shot or few-shot prompting. Here you're asking the AI to think through its answers step by step, to show its working so it can be checked, if you like. For example, you might ask: I've been offered two loans, each for £100,000, repayable over three years.

The loan from Barclays has a 4.5% APR and an arrangement fee of £500.

The loan from Santander has a 4.9% APR and an arrangement fee of £600.

Which has the cheapest overall cost? Think step-by-step and show your working.

When we entered this into *ChatGPT Plus*, it gave us a long set of workings, but that revealed its answer was based on annual payments, not monthly, as you'd expect. To confirm, we asked:

Are you assuming payments are made monthly?

It then went off and calculated the correct figures, answering that the Barclays loan would be around £800 cheaper over the three-year term (it provided more detailed answers than this). This highlights a few key issues:

1. Be as precise as possible with your initial prompt (for example, using "APR" rather than just "interest rate").
2. Get the AI to show its working to have a better chance of spotting mistakes, misassumptions or hallucinations (stuff it's made up).
3. Challenge the AI on its methodology, so you can be as sure as possible that the answers you're getting are correct.

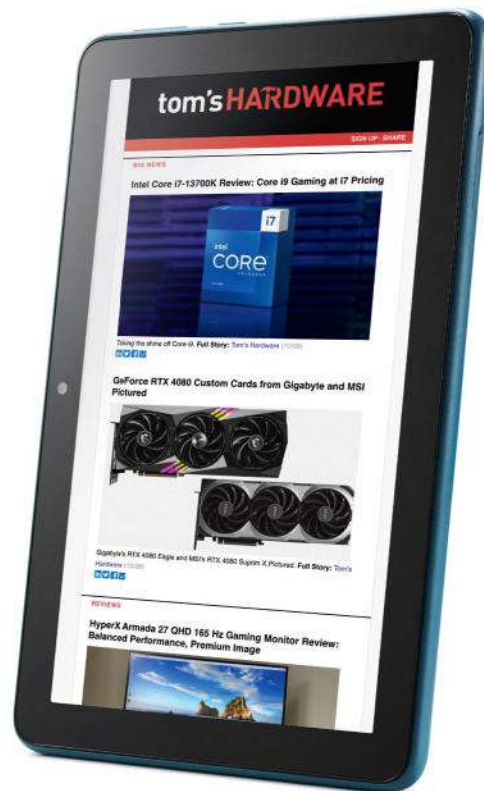
# tom's **HARDWARE**

**GET ALL THE ESSENTIAL BREAKING NEWS FOR THE TECH ENTHUSIAST!**

No matter if you're building a PC, buying a laptop or learning about robots, Tom's Hardware has all the comprehensive knowledge you need.



**Scan & Subscribe  
for free!**



## **GET READY TO BECOME A TRUE PYTHON EXPERT**

Make Python work for you with tutorials on coding with Django, Flask, Pygame and even more useful third-party frameworks.



Ordering is easy. Go online at:

**magazinesdirect.com**

Or get it from selected supermarkets & newsagents



# HotPicks

OBS Studio » Moosync » PhotoCollage » Collector  
 » Patool » Warpinator » Croc » FreeDoom  
 » LibreMines » QOwnNotes » S-tui



## Mayank Sharma

was bitten by a radioactive download manager, which is why he can fearlessly venture into obscure repos and rummage through bleeding-edge software.

### DESKTOP STREAMING

## OBS Studio

Version: 30.0.2

Web: <https://obsproject.com>

**T**here are lots of good options for recording screencasts, but if you want to live-stream your desktop, one of the best options is *OBS Studio*. The app works with all the major online streaming providers, such as Twitch, YouTube Live, Facebook Live and dozens more.

The project hosts a PPA for Ubuntu users, but it's easiest to use the official Flatpak. If your distro is configured to install Flatpaks, you can install *OBS* with **flatpak install flathub com.obsproject.Studio**. Once installed, you can run it from your distro's application launcher or with **flatpak run com.obsproject.Studio**.

When you launch the app, it displays an Auto-Configuration Wizard. This helps determine the best settings depending on your purpose for using the app: streaming, or offline recording. You can rerun the wizard from under the Tools menu.

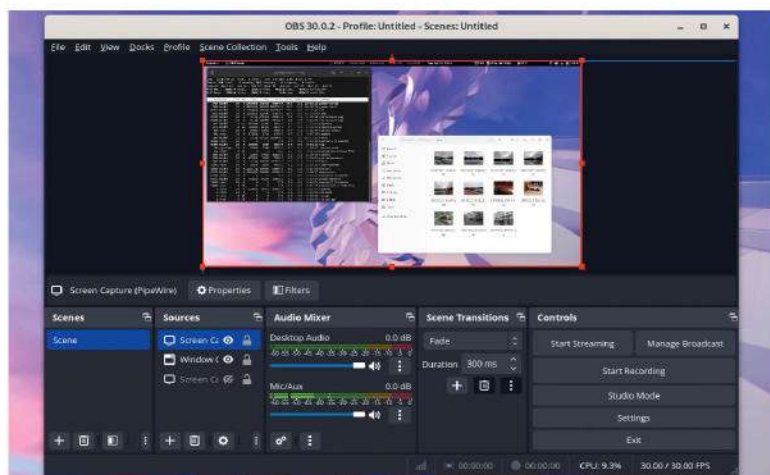
The next step in the wizard is Video Settings. Here you must select the resolution of the recording, with several resolutions available in the pull-down. It's usually a good idea to leave the FPS at the default value unless you have a reason to scale it down.

Next you're asked to select a streaming service. Virtually all of them ask you to add a streaming key. The process for fetching these depends on the service.

When done, head to the Scenes panel, which is populated with a single scene. You can rename it, and add other scenes depending on your streaming plan.

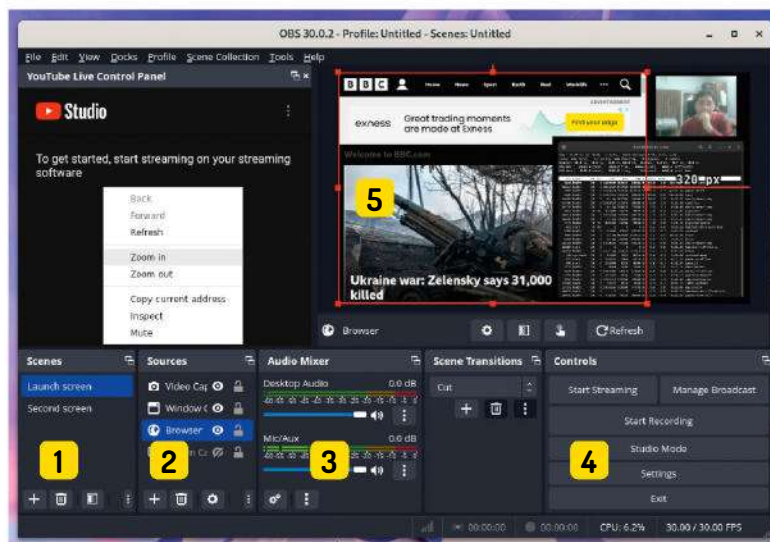
Then switch to the Sources panel to set up the scenes' source. Click the + button to browse through the list of available sources, which includes everything from webcams to audio devices, screen capture tools and dozens more choices. Each source displays its own set of options.

The app already uses your default audio device and microphone as output and input devices, but you have to jump through the other panels to adjust the volume balance and define transitions. When you are done, hit the Start Streaming button.



If you've never used it before, OBS Studio might seem a little daunting at first, but the app has ample documentation to help you get started.

### LET'S EXPLORE OBS STUDIO



#### 1 Scenes

You can set up a stream to have multiple scenes, each with its own elements. You can then switch between them during the stream.

#### 2 Sources

*OBS* supports multiple audio and video streams, and you can divide them between the scenes as per your requirements.

#### 3 Audio Mixer

Here you adjust the volume balance between all the different audio sources. It's

always a good idea to test these before you start streaming.

#### 4 Controls

When you're ready, hit the Start Streaming and Start Recording buttons to begin capturing footage. It also displays the current FPS and CPU usage.

#### 5 Stream preview

This is how your stream will appear. You can drag and resize all the added sources in your stream as per your requirements.



## MUSIC PLAYER

# Moosync

Version: 10.3.2

Web: <https://moosync.app>

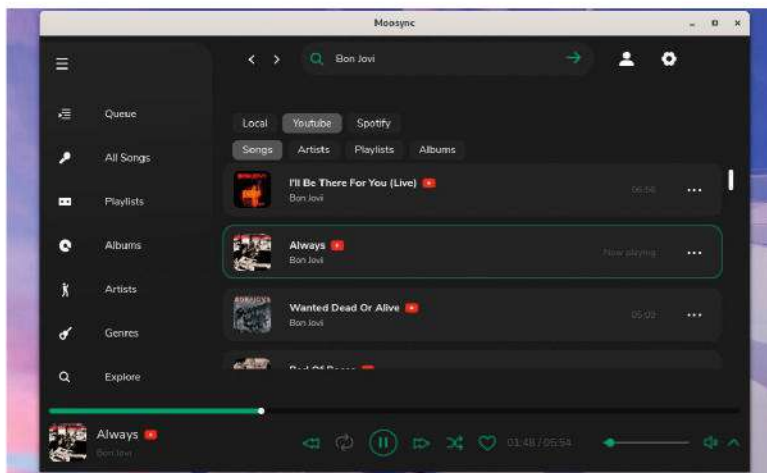
**T**here's no scarcity of music players for the Linux desktop, but what sets *Moosync* apart from its peers is its ability to seamlessly play local tracks and online music.

*Moosync* offers both DEB and RPM precompiled binaries. You can also download the platform-independent AppImage and make it an executable from the file manager or with `chmod +x`.

On first launch, the app starts a setup to help determine the location of your music collection. It defaults to `~/Music`, but you can add and remove folders. You can also use this setup to connect to supported online services, such as YouTube, Spotify and LastFM.

You can optionally log into those services, using IDs and credentials that depend on the service. The good thing, though, is that you can search and play songs from YouTube without logging in.

Once inside the app, the All Songs tab lists all the songs in your local folders. The app also sorts your songs by genre, which you can browse under the



Genres tab. Similarly, you can browse your collection by artist or album under the Albums and Artists tabs.

Double-click on a song to add it to the queue. You can use the three-dot meatballs menu or right-click on a song to bring up more options, such as the ability to play it right away, or add it to the queue to play after the currently playing track.

You also get the option to add a song to a playlist, or to hide or remove it from the library altogether. You can also jump to browse other songs in the same album or by the same artist.

The app has a search bar at the top to help you sort through the music. You can use it along with the YouTube filter to search for music on the service.

Click on the icon in the bottom-left corner to bring up the music queue, where you can ask *Moosync* to fetch the lyrics for the playing track.

## PHOTO EDITING

# PhotoCollage

Version: 1.4.6 Web: <https://github.com/adrienverge/PhotoCollage>

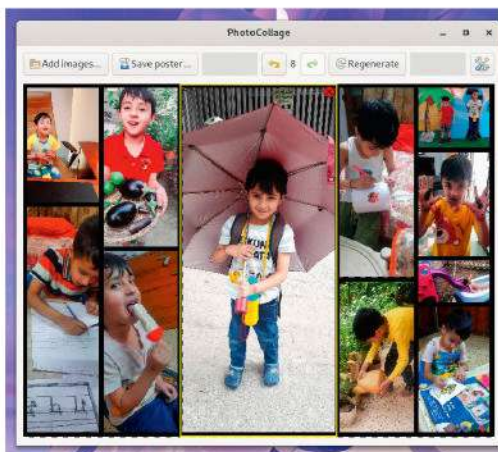
**I**f you have a bunch of photos that you want to stitch together into a collage, you could use venerable tools such as *ImageMagick* or *GIMP*. They can certainly do the job, but it takes time and requires you to have some image-editing chops.

If that's not you, take a look at *PhotoCollage*. The app automates the process and can spit out photo collages with a single click.

It's available in the repos of most popular distros, but the version there would be outdated. However, it doesn't take much to install it from source if you have Python 3:

```
$ git clone https://github.com/adrienverge/PhotoCollage.git
$ cd PhotoCollage
$ sudo python3 setup.py install
$ photocollage
```

The app has a simple interface. Click the Add Images button and select the images you want to add to your collage. And that's it. The app processes the images and then displays a collage.



You can rearrange the images in the collage with simple drag and drop. You can also delete an image from the collage by hovering over it and pressing the X button. Similarly, you can add more images to the collage by using the Add Images button and selecting more pictures.

If you don't like the layout, hit the Regenerate button to ask the app to rearrange the images into a new layout. You can press the Regenerate button any number of times until you get a layout that you find appealing. Helpfully, the app keeps track of all layouts, so you can switch back to any of the older ones using the arrow keys next to the Regenerate button.

When you have a layout you like, hit the Save Poster button to save the image as a JPG, PNG, or WEBP.

*PhotoCollage* defaults to a 800x600 canvas, but you can specify your own dimensions or choose from one of the popular ones, such as A4.

## FILE MOVER

## Collector

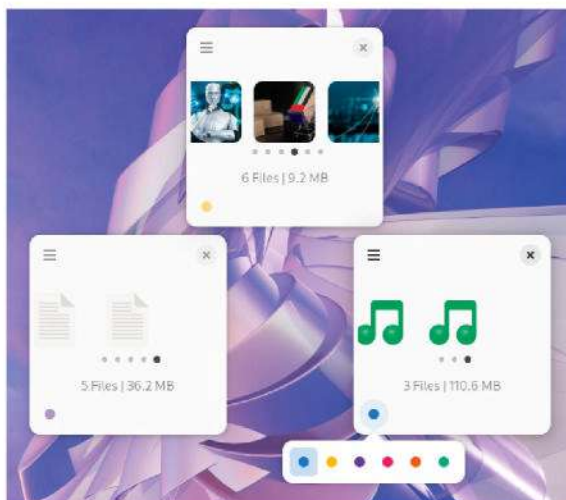
Version: 1.0.1 Web: <https://mijorus.it/projects/collector/>

**Y**our distro's default file manager does a good job of copying and moving files, but it isn't very efficient if your workflow involves ferrying multiple files from all over the filesystem. And this is exactly what *Collector* is designed for. The app functions as a temporary holding space in which you drag and drop files, or paste clipboard contents, web links and more, and then copy them to their destination in one go.

*Collector* is officially distributed as a Flatpak and you can install it with `flatpak install flathub it.mijorus.collector`. When it's installed, the app embeds itself in your Applications menu.

On launch, the app places a small window on the desktop that acts as a temporary holding space for the content you want to move. You can drag and drop all kinds of files, and even paste text and URLs from the clipboard into this window. The window keeps track of all the items dropped into it.

If you want to edit your selection, you can use the arrow keys to jump between pasted items. When you



You can open multiple Collector windows simultaneously and assign each a coloured dot to separate them from each other.

hover over an item pasted into the holding space, you get a three-dot kebab menu. The menu has options to copy the selected item into the clipboard, delete it from the holding area, or open it. To view a file in the holding space, *Collector* depends on an external app installed in your distro. If you have multiple apps that can open that file, the app lets you choose.

When you've finished collecting all the files you want, you can drag them all out with a single click and drop them elsewhere. You can drop them to a folder, or even inside an app or a web page, for instance, to upload them into your online cloud account or as an email attachment.

## ARCHIVE EXTRACTOR

## Patool

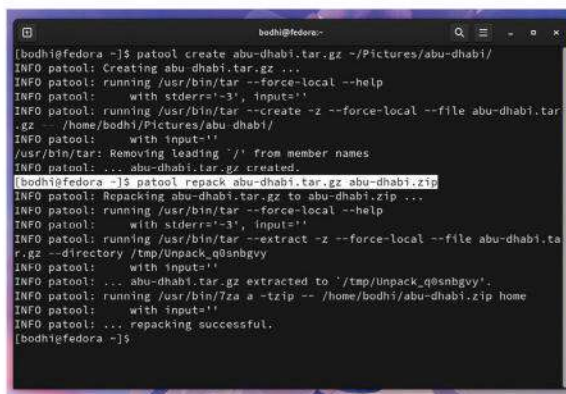
Version: 2.2.0 Web: <https://wummel.github.io/patool/>

**T**AR, TAR.GZ, ZIP and 7ZIP are just some of the endless list of archiving formats you'll run into on a fairly regular basis on the Linux desktop. What makes working with them even more complicated is the fact that they all come with their own tools for managing them. And, of course, these tools all have their own commands and options.

With *Patool*, you don't need to bother to remember the arcane CLI options for each format, because the Python-based utility can handle a wide range of archiving formats, including all the common ones.

So, all you need to do is remember *Patool*'s options to create and manage archives, and the utility automatically translates them for the relevant archiving format behind the scenes.

Before you can use the tool, you first need to install Python 3 from your distro's package manager with `sudo apt install python3` for Ubuntu and `sudo dnf install python3` for Fedora. You can now use the Python *Pip* package manager to install *Patool* with `pip install patool`.



You can also use Patool to repackage an archive into a different format, such as `patool repack original.tar.gz new.bz2`.

*Patool* is now ready to work with your archives. The `extract` command is used to unpack archives – for instance, `patool extract foo.tar.gz`. You can also extract multiple archives, each of which can be a different format, in one go, such as `patool extract firstarchive.zip anotherarchive.tar.gz`.

The utility can also display the contents of the archive without extracting it using the `list` option. Similarly, it can compare the contents of two archives with the `diff` option. The option works even if the archives are in different formats – for instance, `patool diff release-0.1.gz release-0.1.bz2`.

Use the `create` option to build an archive in any of the supported formats, such as `patool create birthdaypics.bzip2 ~/Photos/birthday/*.jpg`. This command creates a BZIP2 archive of all the JPG images in the `~/Photos/birthday` directory.



## FILE TRANSFER

# Warpinator

Version: 1.8.3 Web: <https://github.com/linuxmint/>

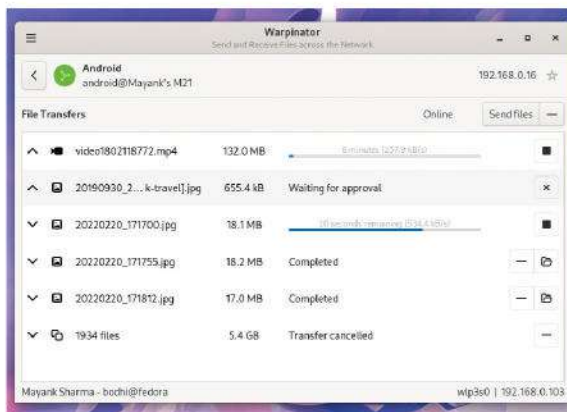
There are tons of ways to send files to remote drives, but shuttling files from one device to another inside the same network isn't always straightforward. But it can be with *Warpinator*.

It's developed by the Linux Mint project, but *Warpinator* is available as a Flatpak, so you can install it on any distribution with `flatpak install flathub org.x.Warpinator`. The app also has unofficial builds for Android, iOS and Windows, which you can find on its website.

By default, the app launches with Secure Mode disabled, as it uses the default group code *Warpinator*. To enable Secure Mode, all you need to do is set a custom group code that is at least four characters long.

Now install *Warpinator* on other distros and devices on your network, and change their group code to the same custom one you've assigned to the first device. At this point, all the computers and devices in your network can see each other.

However, since Linux distros run a firewall, you have to poke holes in it before *Warpinator* can ferry files



Warpinator lets you change the default ports, and it's a good idea to do so. Just make sure you allow traffic through them in your firewall.

through them. Again by default, the app uses the 42000/tcp and 41000/udp ports to transfer files. If you're using Ubuntu, you can allow these ports in its UFW firewall with `sudo ufw allow 42000/tcp` and `sudo ufw allow 42001/udp`.

Similarly, in Fedora use `sudo firewall-cmd --add-port=42000/tcp` and `sudo firewall-cmd --add-port=42001/udp`. In Fedora you have to then make the changes persist across reboots with `sudo firewall-cmd --runtime-to-permanent`.

The Linux computers should now be able to transfer files without any issues. The main interface lists all the other *Warpinator*-equipped devices in your network. By default, the app prompts the recipient to approve the transfer, but this can be overruled from within the app's preferences.

## FILE TRANSFER

# Croc

Version: 9.6.12 Web: <https://schollz.com/tinker/croc6/>

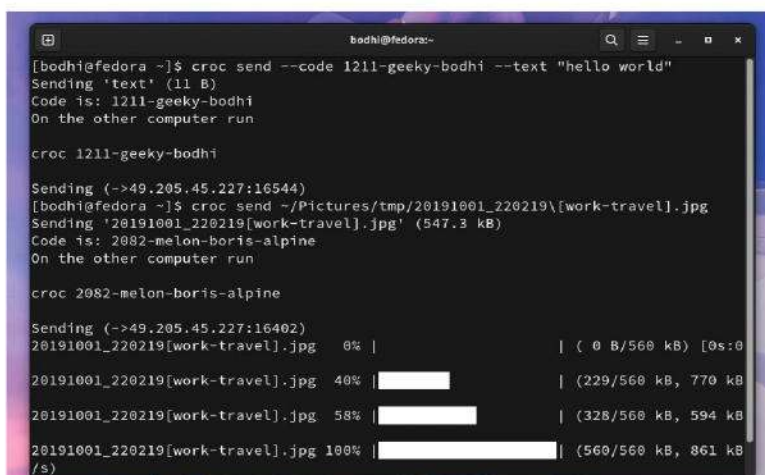
If you are a CLI warrior or don't want to restrict yourself to the local network, *Croc* can ferry files to any computer on the internet.

It is available in the repos of most distros, but it's best to install the latest release from source. This doesn't take much effort and can be achieved with `curl https://getcroc.schollz.com | bash`. This downloads and extracts the latest release in `/tmp`, and then installs it under `/usr/local/bin`, after prompting you for the superuser password. *Croc* is cross-platform and can be installed in Mac OS and Windows as well.

Use the `send` option to send a file, such as `croc send ~/Documents/some-folder`. This displays the size of the transfer, along with a code, which you must send to the recipient.

Then to receive the sent files on another computer, the recipient enters the code with the `croc` command, such as `croc code-phrase`, which begins the transfer.

*Croc* uses the code phrase to establish password-authenticated key agreement (PAKE). This generates a key that ensures the transfer is end-to-end encrypted.



Instead of relying on the utility to generate a code, you can use a custom one with the `--code` option, as long as it is more than six characters long. For instance, `croc send --code [custom-code-phrase] doc1.txt doc2.txt ~/Documents/other-files` assigns the specified custom code to the transfer.

By default, *Croc* asks whether to overwrite a file. The recipient can use the `--yes` and `--overwrite` options to silently overwrite files, such as `croc --yes --overwrite [secret-code]`.

You can also send plain text with *Croc*, such as `croc send --text "hello world"`. This tells the *Croc* utility at the recipient's end to display the sent text in the terminal window.

It might be a CLI tool, but the good thing about *Croc*, besides its security and encryption features, is that it doesn't require any port forwarding.

**FPS**

# Freedom

**Version:** 0.13.0**Web:** <https://freedom.github.io>

**W**e just can't get enough of *Doom*. The first-person shooter genre has grown in leaps and bounds since its first release back in the early '90s, but there's a certain charm to the original that just keeps pulling fans back for more. The community has been churning out ports and mods ever since the *Doom* source code was released.

The Freedom project is one such enterprise that produces game data files (or WADs) for *Doom*-compatible gaming engines. The project is a collection of three games.

Firstly, there's *Freedom: Phase 1*, which is made up of four episodes, each with over half a dozen levels. It offers interesting gameplay with tons of mazes and enemies to battle.

Then there's *Freedom: Phase 2*, which has a single episode spanning over three dozen levels. It builds on *Phase 1*, and adds more weapons and monsters, and boasts even more enthralling gameplay.

The third is *FreeDM*, which is a monster-free set of levels, made specifically for multiplayer deathmatches.



You can use FreeDM to engage your mates, but the exact mechanism for hosting a multiplayer game depends on the source port.

The project distributes all three variants of the game as WAD files, which means they contain all the artwork, levels, music and sound effects that make up the game itself. To actually play the game, you need to plug the WAD files into one of the dozens of *Doom* source ports, such as *GZDoom* or *Crispy Doom*.

If you've been playing *Doom* for as long as us, this shouldn't be a problem. But if you want to experience *Freedom* without the hassle of first setting up a source port, you can grab the games from Flatpak, which come bundled with *GZDoom*.

Head to the terminal and type `flatpak install flathub io.github.freedom.Phase1` to install *Phase 1*, `flatpak install flathub io.github.freedom.Phase2` to install *Phase 2*, and `flatpak install flathub io.github.freedom.FreeDM` to install the multiplayer deathmatch variant.

**PUZZLE**

# LibreMines

**Version:** 2.0.0 **Web:** <https://github.com/Bollos00/LibreMines>

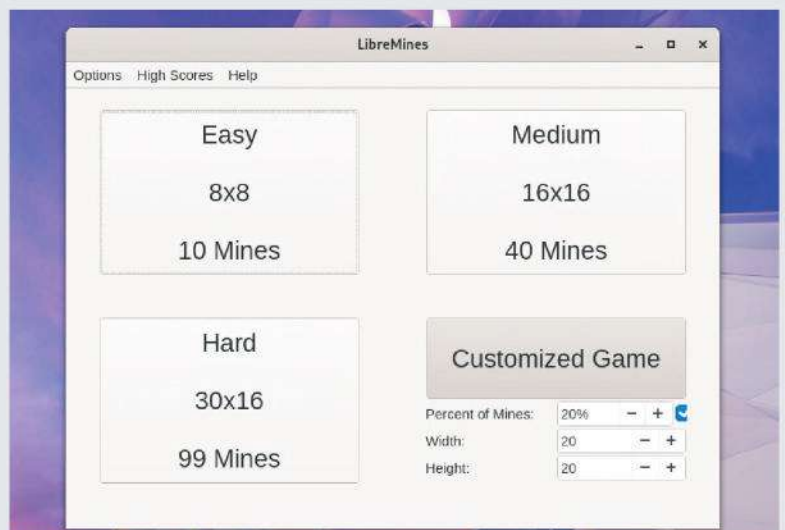
**A**nother '90s classic, *Minesweeper* doesn't have the same kind of fan following that *Doom* has with FPS players, but it did help popularise the puzzle genre. *LibreMines* continues that legacy with its cross-platform Qt-based clone.

The game is available in the repos of many distros, but it's best to pull it from Flathub with `flatpak install flathub io.github.Bollos00.LibreMines` to ensure you have the latest release.

On launch, the game offers three predefined modes: easy, medium and hard. The three levels have differently sized grids and different numbers of mines. There's also a fourth option to create a customised game, where you can choose the grid size and the number of mines, specified as a percentage of the grid.

The gameplay is the same as the original. The grid has clickable tiles with hidden mines scattered through the board. Your objective is to clear the board without detonating any mines, with the help of clues.

When you click on a cell that isn't hiding a mine, it displays a number that indicates the number of mines



diagonally and/or adjacent to it. You use this to plot your next click. You can also flag a cell, which remains unopened, to indicate a potential mine location. To win a game, you must open all non-mine cells without detonating a mine.

You can play *LibreMines* with both the keyboard and the mouse. Use a left-click to open a cell, and a right-click to flag or unflag one. To activate the keyboard controller, press any of the A, S, D or W keys, and then use them to move through the board. Press O to toggle a cell, and P to flag or unflag it. The Esc key exits the keyboard controller mode.

LibreMines has several customisable options, including all sorts of light and dark themes, as well as sound effects.

## NOTE TAKING

## QOwnNotes

Version: 24.2.5

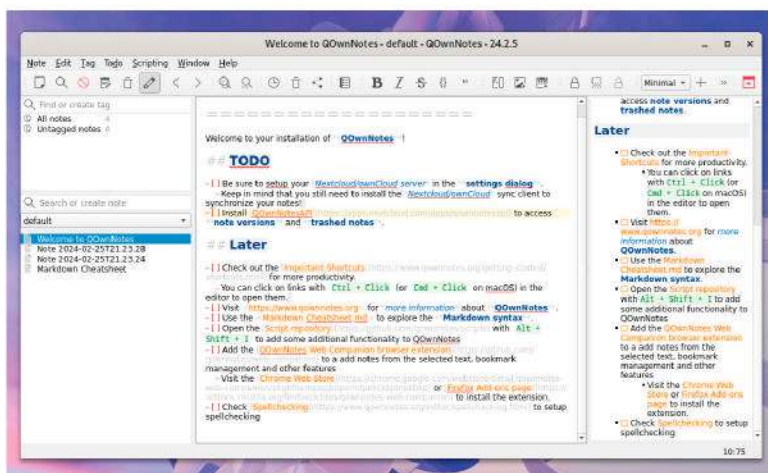
Web: [www.qownnotes.org](http://www.qownnotes.org)

**L**inux users are spoilt for choice when it comes to note-taking apps. *Joplin* and *Trilium* are some of the best in the game, but if you're concerned about the appearance of your notes, take a look at *QOwnNotes*. One of the best things about this app is its support for Markdown.

*QOwnNotes* is available in the official repos of all the popular distros. The project also puts out an AppImage of the app that you can download from its GitHub page, and make it an executable from the file manager or with `chmod +x`.

On first launch, the app prompts you for the location where you want to store your notes, which by default is `~/Notes`. Next you have to select from one of five layout options. You can scroll through each to read a description and see a handy screenshot before making your choice.

Apart from the single-column view, all other layouts have a split-pane interface. The minimal view is a good starting point for most users, and you can then pile on panels when you get the hang of the editor.



You can jump straight into the editor after you've selected a panel layout. But if you have a Nextcloud or OwnCloud instance, you can continue working through the wizard to hook up *QOwnNotes* with your file-sharing server.

*QOwnNotes* is fairly intuitive, especially in the minimal view. New notes are added to the searchable list of notes in the sidebar. After you've typed in your note, you can preview it in the view-only preview panel.

When satisfied, you can export a note in HTML, PDF or Markdown. A nice touch is its ability to export PDFs that can either be Markdown or formatted files that don't show the underlying Markdown.

While *QOwnNotes* supports Markdown and has a cheat sheet to get you started, it does provide the usual formatting buttons to style the text.

## STRESS TEST

## S-tui

Version: 1.1.6 Web: <https://amanusk.github.io/s-tui/>

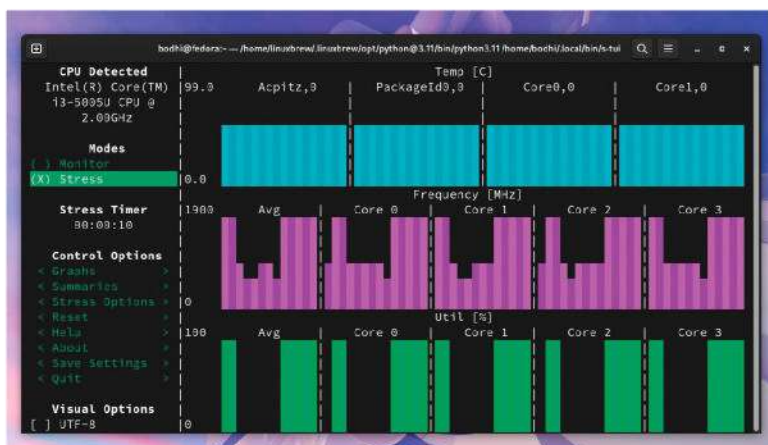
**L**inux doesn't have many stress-testing tools, but it isn't surprising that one of the ones it does have works on the CLI. The *Stress-Terminal UI*, or *S-tui*, can maximise your hardware as per your whims, and also doubles up as a nice CLI-based system monitor.

It is available in the repos of most distros. However, the developers recommend installing it via Python's *Pip* package manager to get the latest release.

If you don't have Python, you can easily pull it from your distro's official repos, either with `sudo apt install python3` (if you're running Ubuntu) or with `sudo dnf install python3` (if you're running Fedora). You can then install *S-tui* with `pip install s-tui --user`.

You also need to equip *S-tui* with an actual stress tester, such as the unimaginatively named *Stress* utility. Again, this is available in the official repo of your distro and is just a `sudo apt install stress` (Ubuntu) or `sudo dnf install stress` (Fedora) away.

You're now all set. Unlike many CLI utilities, *S-tui* doesn't need to be configured before launch, as most



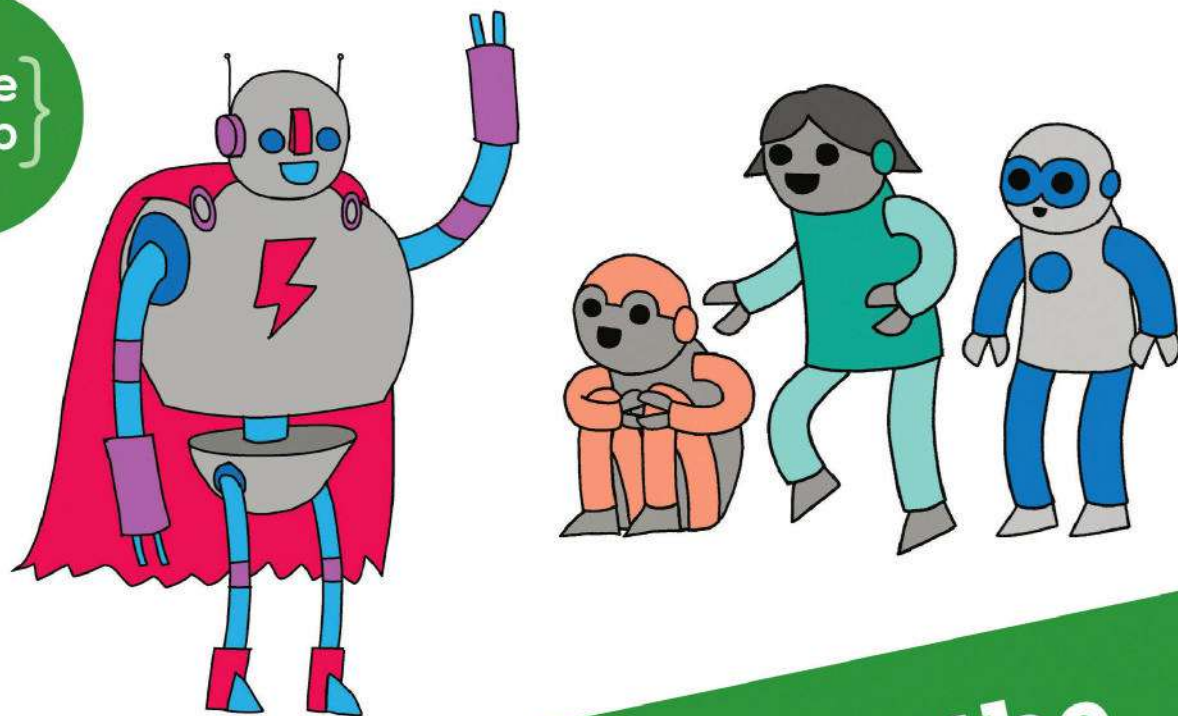
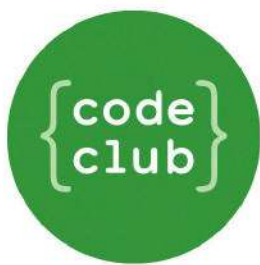
of the stress-testing parameters can be set inside the utility. So, go ahead and type `s-tui` to fire it up.

By default, the app launches in the monitoring mode, where it displays hardware details about your computer along with some other parameters. The tool's tweakable parameters are all nestled in the left column. You can use the *Graphs* and *Summaries* sections under *Control Options* to toggle any of the monitored hardware.

Similarly, you can adjust the stress-testing features from the *Stress Options* section. Head in there to review the default options, and customise them as per your requirements. When you're done, scroll to the top of the interface, and toggle the *Stress* option under the *Modes* section. **LXF**

Unlike many CLI utilities, you can navigate the *S-tui* interface and tweak various parameters using the mouse.





**Can you help inspire the  
next generation of coders?**



**Code Club** is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!  
So to find out more, join us at **[www.codeclub.org.uk](http://www.codeclub.org.uk)**

## PYTHON

# Solve word puzzles with clever code

Living conundrum **Matt Holder** discovers how to use Python to improve his puzzle-solving skills when watching Countdown.



OUR  
EXPERT

**Matt Holder** is an IT professional of 15 years, Linux user for over 20 years, home-automation fan and self-professed geek.

**W**e are going to create a program in Python that can solve word puzzles. In the UK, we have a long-running game show called *Countdown*, which has three types of round:

- **Numbers:** One contestant picks six unseen cards, each containing a number. A random number is then generated by one of the hosts and the two contestants have 30 seconds to use the numbers to calculate the generated total. It is not always possible to reach the generated value, so the nearest gets the most points.

- **Words:** One contestant picks nine unseen cards from two piles – vowels and consonants. The contestants then use the 30-second countdown period to find the longest word possible from those letters. If both contestants find the same length word, they both get points equal to the length of the word, but if one gets a longer word, only that contestant gets the points.
- **Conundrum:** The final round of the quiz is called the conundrum. The contestants see a nine-letter word with its letters jumbled up, and they use the countdown period to work out the mystery word. The winner of this round is awarded 10 points.

With that introduction done, let's look at what we are doing in this article. We are taking *Countdown*'s words round and generating a piece of Python code to find words from a series of letters. The program will be capable of using letters you enter (should you wish to play along while watching the TV show) as well as generating a random set of letters. Once words have been discovered, we will be able to display them to screen and write them to a file as well.

So, let's begin. The first thing to do is create a new directory to start the project. Open your favourite IDE and create a file called **countdown.py**. If needed, install any Python add-ons for the IDE and set the language of the program you are writing to Python. When



The Countdown show's set looks very similar to the one shown in the official game.

running the program, if any libraries are missing, they can be installed using the command structure below. The rich library is the most likely one to be missing and it allows the colouring of text in the console.

```
$ pip install rich
```

The first thing we do in the program is import any libraries and declare a new class, which will be used to load the word file, generate combinations of letters and test against the word list.

```
import hashlib
import random
import itertools
from rich.prompt import Prompt
from rich.status import Status
from rich.console import Console
import sys
import time
import os
from math import floor

class Countdown:
    def __init__(self, letterNumber=9):
        self.wordDict = {}
        self.discoveredWords = []
        self.letterNumber = letterNumber
```

### QUICK TIP

The full source code can be downloaded from <https://github.com/mattmole/LXF-Countdown-Word-Solver>



```
self.generatedLetters = ""
self.wordPrefixDict = {}
```

In the first 10 lines of this listing, we import the libraries we're using. The rest of the listing is used to create a class, which is called **Countdown**. The `__init__` function is then defined, which is used to initialise any instances of the class, which are created. This function takes a single argument, which is the number of letters to generate. This has a default value of 9, but this can be overwritten at any time during the program's running.

Finally, we set up some variables. The **wordDict** dictionary is used to store the words that are loaded in from the word file. This uses the sha256 hash of the word as the dictionary's key and the value is the word itself. This may seem a strange way of loading in the list, but the reason will become apparent.

The **discoveredWords** variable is a list, which stores any words that are found. The **letterNumber** variable stores the number of letters to generate, **generatedLetters** stores the letters we generate either randomly or by entering a list, and **wordPrefixDict** is used to cut down on processing time by ignoring any combinations that contain sequences that are not in any words, such as zzz.

```
def generateHash(self, letters):
    m = hashlib.sha256()
    m.update(letters.encode('utf-8'))
    return m.hexdigest()
```

The **generateHash** function is used multiple times within the program to generate the sha256 hash of the input to the function. We generate an instance of the sha256 method from the hashlib library. We then encode the letters to utf-8, update the hashing object and then return the **hexdigest** value. This value is returned and used elsewhere.

```
def loadFile(self, wordFile="words_alpha.txt"):
    with open(wordFile, 'r') as file:
```

The **loadFile** function is used to load the dictionary words in from a file, which can be downloaded from the internet. The validity of results is purely down to this list, and it may take some time to find a really good one. The first thing we do is define the function and set a default **wordFile** argument of **words\_alpha.txt**. If the function is called without an argument, the default is used, otherwise the file referenced by the argument is opened. Next we use **with** functionality, which is a way of performing tasks while the file is open. Once the processing is complete, the file is automatically closed.

The **with** keyword is used, along with opening the file and the handle called **file** is used to reference the open file.

```
    for line in file:
        word = line.strip()
        if len(word) <= self.letterNumber:
            self.wordDict[self.generateHash(word)] = word
            prefixLetters = word[0:floor(1/2*self.
            letterNumber)].lower()
            prefixHash = self.
            generateHash(prefixLetters)
            if prefixHash not in self.wordPrefixDict:
                self.wordPrefixDict[prefixHash] = ""
```

We next process each line of the file using a **for** loop and we strip any return characters from the text in question. We only process words in the word list that



The GUI application can be seen here. Note, the word list used may not be the best...

are the same length as the number of letters we are testing against or shorter.

When the number of letters requested is very high, we need to think about ways of speeding up the processing because it would take a long time to complete. For each word that meets our condition, we generate the sha256 hash and add it to the **wordDict** dictionary. As can be seen in the previous code, the hash is used as the key and the word as the value. Next, within this function, we generate a prefix, which we can use to test if the generated letters contain the valid beginning to the word. For example, zzz would not be valid, so we would not want to test this generated combination against the word list.

The value is calculated as the nearest integer below the value of half of the number of letters. For example, should 11 letters be picked, the first five would be used

### QUICK TIP

The rich library is incredibly powerful and can be learned about at <https://github.com/Textualize/rich>

Some of the code used to write the GUI can be seen here.

```
a = Countdown()
app = App(title="Countdown solver", layout="grid")
welcomeText = Text(app, text="Welcome to the Countdown word solver app!", grid=[0,0,4,1])
enterLettersButton = PushButton(app, command=enterLetters, text="Enter letters", grid=[0,1])
generateLettersButton1 = PushButton(app, command=generateLetters1, text="Generate letters", grid=[1,1])
findWordsButton = PushButton(app, command=findWords, text="Find words", grid=[2,1], enabled=False)
displayWordsButton = PushButton(app, command=displayWords, text="Display words", grid=[3,1], enabled=False)
numLettersTextBox = TextBox(app, text="9", grid=[0,2,2,1], enabled=False, visible=False)
generateLettersButton2 = PushButton(app, command=generateLetters2, text="Generate", grid=[2,2,2,1], visible=False)
lettersTextBox = TextBox(app, text="", command=setLetters, width=40, grid=[0,3,4,1], enabled=False, visible=False)
discoveredWordsText = Text(app, text="Discovered words", grid=[0,4,4,1], visible=False)
discoveredWordsListBox = ListBox(app, width="fill", grid=[0,5,4,4], enabled=False, visible=True, items=["*40"])
app.display()
```

## » MULTI-PURPOSE LOGIC

The powerful part of writing the initial code in the way we did is that the logic can be reused for many different purposes. The code could be used for operation from the CLI, from a GUI, or even via a command-line argument, where the operation does not require menu items to be selected at all. This box discusses the GUIZero library, which utilises the Python TK interface, but contains a lot of boilerplate code hidden behind the library's methods. The code in the screenshot (above) shows how the library can be used to create widgets on the screen to allow the library to be used.

In this sample, a copy of the *Countdown* library is instantiated, before a GUIZero app is created. Following this, text boxes, labels, buttons and list boxes are defined. Note how some of the widgets are hidden by default, so they can be enabled at a later time.

The application, as shown in the GUI screenshot (top), utilises a very similar methodology to that of the command-line application. Letters can either be added or generated randomly, with words discovered from combinations and then displayed on the screen.





to build the prefix dictionary. We again use a dictionary, with the hash of the prefix as the key, but we do not need the value, so set this to an empty string.

```
def genRandomLetters(self):
    vowels = "aeiou"
    consonants = "bcdfghjklmnpqrstvwxyz"
    numConsonants = random.randint(4, self.
letterNumber - 3)
    numVowels = self.letterNumber - numConsonants
```

## » CREATING AN API

As seems to be the trend over the past few years, absolutely everything needs an API. This is no different with our *Countdown* program and the excellent FastAPI library makes it incredibly easy to create one. The FastAPI library and supporting web server can be installed by opening a terminal and entering the following:

```
$ pip install fastapi
$ pip install uvicorn[standard]
```

The documentation (see screenshot below) is created by the FastAPI library. The code below is used to register the endpoint to generate a random set of letters (see GitHub for a fuller example):

```
from fastapi import FastAPI
from countdown import Countdown
```

```
a = Countdown()
app = FastAPI()
```

```
@app.get("/generate/{numLetters}")
def generateWord(numLetters):
    a.letterNumber = int(numLetters)
    a.genRandomLetters()
    return {"numLetters": numLetters, "generatedLetters": a.
generatedLetters}
```

In this code sample, we import libraries and create instances of each. We then decorate a function with one of the FastAPI decorators – this essentially adds functionality to the function we create. The function takes an argument, which contains the number of letters. We then use the **Countdown** class to set the number of letters and generate a random set. We then return information using Python dictionaries. Running the uvicorn web server allows this to be tested:

```
$ python3 -m uvicorn countdownAPI:app --reload
```

Now, open your web browser and navigate to **http://localhost:8000/generate/8** and you will see an eight-letter random string is generated and returned. This can then be extended to other methods as well.



FastAPI uses swagger to generate documentation from the Python code.

```
randomLetters = ""
for i in range(0,numVowels):
    randomLetters += random.choice(vowels)
for i in range(0,numConsonants):
    randomLetters += random.choice(consonants)
self.generatedLetters = randomLetters
return randomLetters
```

In the above code, we generate a list of random letters. *Countdown*'s rules say there should be at least three vowels and four consonants, which we handle here. It could be better calculated when the number of letters increases, but that's left as an exercise for you.

In this function, we first add a list of vowels and consonants to appropriately named variables. We then calculate the number of consonants by using the random library. This should be at least four and less than the number of letters with three subtracted, which accounts for at least three vowels. The number of vowels is then calculated by taking the number of consonants and subtracting it from the number of letters selected. What follows is the creation of a variable to store our random letters and two **for** loops, one adding the random choices of vowels to the **randomLetters** dictionary and the second adding the consonants. Finally, the function returns the value of the selected letters.

```
def genLetterCombinationsAndTest(self):
    self.discoveredWords.clear()
    for i in range(2,self.letterNumber+1):
        for combination in itertools.permutations(self.
generatedLetters,i):
            prefixLetters = "".join(combination[0:floor(1/2*
self.letterNumber)])
            prefixHash = self.generateHash(prefixLetters)
            if prefixHash in self.wordPrefixDict:
                constructLetters = "".join(combination)
                self.testWord(constructLetters)
```

In the above function, we take the generated letters and calculate all possible permutations, which we can then test against the word list, which we have loaded from the file. Within the function, we first of all clear the list, which contains any discovered words. We do this so we can account for any repeat runs of the discovery process. We then use a **for** loop so we can test letter combinations of different lengths. The **range** function returns a list of numbers within the range. This is set to be from two to the length of the letters plus one. The additional number is needed because the range function is exclusive of the upper number.

We use the permutations method from the *itertools* library to generate all two-letter, non-repeating letter combinations from the input. Once the loop has completed the first time, it repeats for three, four and so on letter combinations. The first half of the letters of the combination that is generated is then checked against the **wordPrefixDict** to determine if it is a valid beginning to a word or not. If the check is successful, the combination is converted to a string (*itertools.permutations* returns a tuple of letters) and then passed to the **testWord** function.

```
def testWord(self,letters):
    hashedLetters = self.generateHash(letters)
    if hashedLetters in self.wordDict:
        if self.wordDict[hashedLetters] not in self.
discoveredWords:
```

```

        self.discoveredWords.append(self.
wordDict[hashedLetters])
    return self.wordDict[hashedLetters]
else:
    return -1

```

The final method we have written in the class is called **testWord**. This function takes the letters passed to it, hashes it and then checks if it is in the **wordDict** dictionary. If this check is successful, the discovered word is added to the **discoveredWords** list. The function then returns the word. If the hash is not in the word list, it returns -1.

With the logic for the main part of the program written, we need to instigate it and call the various methods. At this point, it would be possible to write a program using a GUI framework, such as GTK, QT or Kivy, the command-line *nurses* library or writing a simple interface using **while** loops and **if** statements.

```

if __name__ == "__main__":
    console = Console()
    a = Countdown()

    carryOn = True
    while carryOn:
        console.clear()
        firstLoad = False
        console.print("1) Generate a random set of letters")
        console.print("2) Find words from the set of
letters")
        console.print("3) Display any found words")
        response = Prompt.ask("Enter number from the
list above")
        if response == "3":
            if len(a.discoveredWords) == 0:
                console.print("No words could be found")
            else:
                console.print("Discovered words...")
                for word in a.discoveredWords:
                    console.print(f"\t{word}")
                Prompt.ask("Return to main menu")
        elif response == "2":
            a.loadFile()
            with Status("Discovering dictionary words") as
status:
                a.genLetterCombinationsAndTest()
        elif response == "1":
            numLetters = Prompt.ask("Enter number of
letters")
            intNumLetters = int(numLetters)
            a.letterNumber = intNumLetters
            a.genRandomLetters()
            selectedLetters = a.generatedLetters

```

The first thing we do in this code sample is to test whether the program is being called directly or imported. This is helpful so that if it is imported from elsewhere, it doesn't call this part of the code. On the next line, we create a copy of the console method, which we use for the display of the program. On the next line, we create a copy of the Countdown library and assign it the variable name of **a**.

The rest of the program utilises a **while** loop, so that after any stage of the process has run, the menu shows

```

class Countdown:
    def __init__(self, letterNumber=9):

        self.wordDict = {}
        self.discoveredWords = []
        self.letterNumber = letterNumber
        self.generatedLetters = ""
        self.wordPrefixDict = {}

    def generateHash(self, letters):
        m = hashlib.sha256()
        m.update(letters.encode('utf-8'))
        return m.hexdigest()

    def loadFile(self, wordFile="words_alpha.txt"):
        with open(wordFile, 'r') as file:
            for line in file:
                word = line.strip()
                if len(word) <= self.letterNumber:
                    self.wordDict[self.generateHash(word)] = word
                    prefixLetters = word[0:floor(1/2*self.letterNumber

                    prefixHash = self.generateHash(prefixLetters)
                    if prefixHash not in self.wordPrefixDict:
                        self.wordPrefixDict[prefixHash] = ""

```

Some of our code can be seen here, which is nicely highlighted by VS Code.

again and allows the next stage to be completed. The **carryOn** variable is first set to **True** and this is tested each time the loop runs. If we want to exit the loop, we simply need to set the variable to **False**. Once inside the loop, we clear the console and display the list of choices, before requesting a choice to be selected by using the **Prompt.ask** method.

Any text input is added to the response variable. When the number three is entered, we test to ensure that words have been found. If not, an error is shown, and otherwise the words are printed to the screen. If the number two is entered, we first of all load the word list into memory and assign to the variable within the class. We then call the **genLetterCombinationsAndTest** method of the **Countdown** class to test for valid words. Finally, if number one is entered, we ask how many letters should be generated, set the **letterNumber** variable within the **Countdown** class and generate the correct number of random letters. This **genRandomLetters** method automatically sets the internal variable accordingly.

For space reasons, some functionality has been removed from the code listing that has been published here. Please see the full source code on GitHub (see *Quick Tip*, page 90) for the full version as well as a GUI application. We hope that you have enjoyed this tutorial and can use the techniques used to develop your own similar algorithms. **LXF**

#### QUICK TIP

GUIZero can be used to create GUIs and is quick and easy to use: <https://lawsie.github.io/guizero/>

```

*****
Welcome to the countdown word solver
*****

1) Enter a set of letters
2) Generate a random set of letters
3) Find words from the set of letters
4) Display any found words
5) Write discovered words to file
6) Quit
Enter number from the list above: █

```

The command-line interface (CLI) version of the program can be seen here.

» **GET MORE JUMBLES OF WORDS** Subscribe now at <http://bit.ly/LinuxFormat>



# Pixel-perfect Asteroid collision detection in C

Did those pixels touch or just pass by? **David Bolton** demonstrates a method of checking collisions with lots of objects on screen.



**OUR  
EXPERT**

**David Bolton** played *Asteroids* so much in his youth that he still dreams that he's surrounded by drifting white shapes.

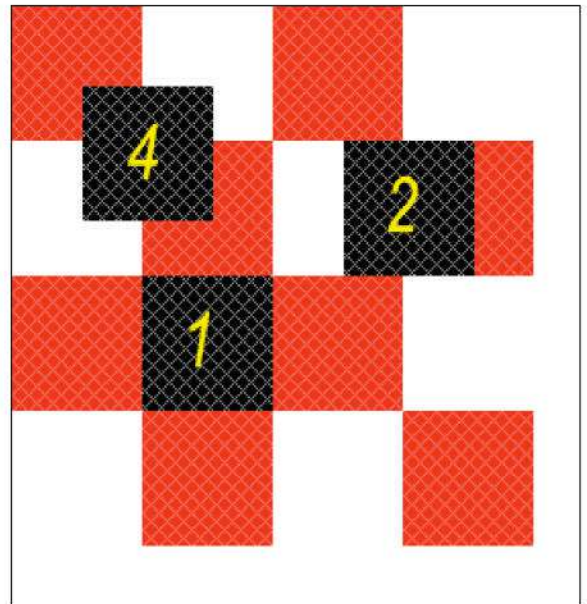
**T**he *Asteroids* videogame uses a high-speed collision detection method to check when bullets are near or might hit asteroids or enemy ships, or when an asteroid might hit your own ship. This article explains how the collision detection can be achieved and at speed.

The original Atari *Asteroids* game used wireframe graphics drawn by a dedicated graphics chip. But if we're going to make it today, we'll be using solid – that is, raster – graphics. This version of *Asteroids* has graphic images for asteroids, a player's ship, bullets, explosions and alien ships.

Accurate collision detection is important – it's not much fun if your ship blows up when a bullet flies past but doesn't actually touch it.

## Graphic impact

First, we need to discuss the graphics. There are 24 each of four different asteroid sizes: 35, 70, 140 and 280 pixels square, with one image for each size containing 24 rotations. Asteroids and the player's ship can rotate in 15-degree increments, so there are 24 images in a PNG in two rows of 12. Graphic 0 is facing



Red and white grid cells with black objects overlapping and showing how many cells each one overlaps.

## » ASTEROIDS GAME IN C

There are about 2,200 lines of C in four source files and three headers. The code compiles and runs on an x86 Linux or Raspberry Pi. Note there is some Pi-specific code but that is only enabled when running on a Pi system and not called on other Linux boxes.

First get the source code and files from GitHub (<https://bit.ly/lxf314astro>). Create a folder called **asteroids** and fetch everything from the GitHub **asteroids** folder. You'll also need the VS Code ZIP (<https://github.com/David-H-Bolton/Projects/blob/main/vscode.zip>) file. Download and unzip the **.vscode** folder into the **asteroids** folder. First

make sure you have the dev SDL libraries installed. Run these four commands in a terminal:

```
$ sudo apt install libsdl2-dev
$ sudo apt install libsdl2-image-dev
$ sudo apt install libsdl2-mixer-dev
$ sudo apt install libsdl2-ttf-dev
```

To compile **asteroids.c**, GCC or Clang will do. There is a supplied **tasks.json** to simplify things but it is preconfigured for Clang 14. If your Clang version is different, edit **tasks.json** and change the args command to your version:

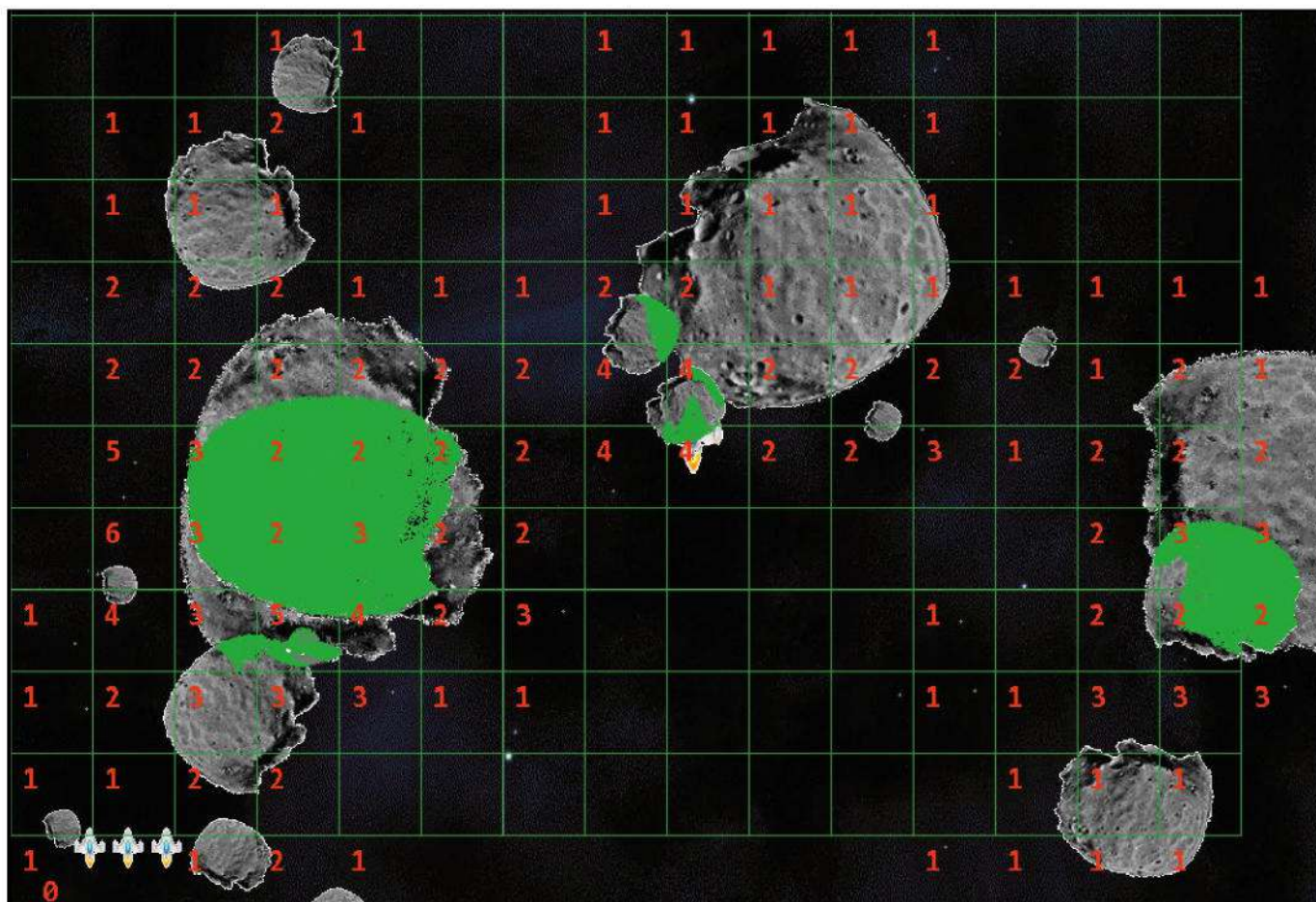
```
"command": "/usr/bin/clang-14",
```

The folders **images**, **masks** and **sounds** must be located in the **asteroids**

folder. In *Visual Studio Code*, the **asteroids** folder should be the Work folder you select when you click on the Explorer tab. The **.vscode** folder should be under the **asteroids** folder as well.

After compiling, the game is started by double-clicking the compiled **asteroids** executable or with the command: **./asteroids** in a terminal.

The keys are **q** and **w** to rotate the ship, **Space** to fire the gun, with a maximum of 16 bullets on screen at once. The **s** key puts up a shield but this counts down as long as **s** is pressed, and **j** does a hyperspace jump. You can't jump again for a couple of seconds after.



north, then graphic 1 is at a rotation of 15 degrees, and so on. The explosion graphics don't move or rotate, so we won't discuss them.

The player's ship fits inside a 64x64 cell. The game's playing area is 1,024x768 pixels. You'll see most of these defined by `#defines` in lines 32-41 of `asteroids.c`. A bit lower down at lines 51-66 are the numbers of things. The maximum number of the four asteroid sizes are 4, 4, 5 and 5. These are defined in `levels.c`, which has an array of structs `levels[50]`. When the larger asteroids are hit, they split up into eight smaller

asteroids, so the number of asteroids on screen can increase very rapidly.

### Figuring out collisions

At any point in the game, there can be a lot of moving objects on screen and we need to figure out a lot of collisions 50 times a second. If there are 50 objects on the screen at the same time, then that's 50 possible collisions. However, they are scattered across the screen, so it's unlikely that more than three or four are ever going to be colliding in any particular part of the screen.

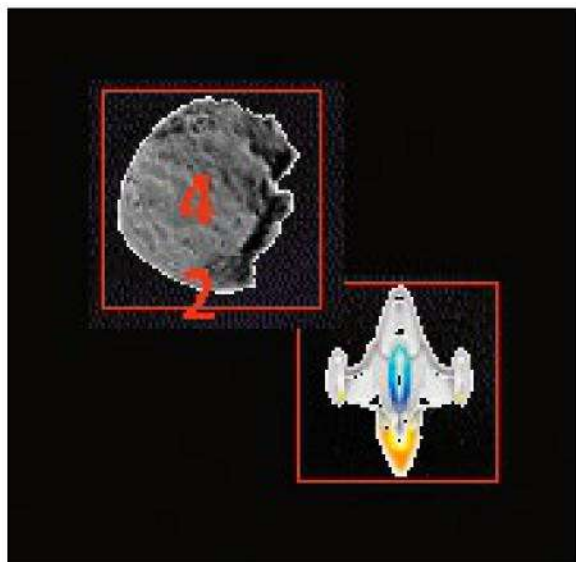
So, we start by dividing the screen into 64x64 cells. There are 192 of these – 16 across and 12 down. Every frame, as the objects move, we calculate which of the cells each object overlaps. The player's ship at 64x64 pixels might be completely in one cell but it is more likely to be across two or four cells. The biggest object – the 280x280-pixel asteroid – can overlap 5x5, 5x6, 6x5 or 6x6 cells.

The black squares represent a 64x64 object overlapping the red and white grid squares. The number shows how many grid squares are overlapped by each object. Bullets are 3x3 pixels and most of the time only overlap one square but they can also overlap two or occasionally four squares.

Each cell has a `numobjects` count and an array of pointers:

```
struct Cell {
    ptrarray ptrs;
    int numobjects;
};
```

Screenshot with ShowCells uncommented and SHOWOVERLAP active.



Bounding boxes surrounding game objects overlapping plus debug text.

### QUICK TIP

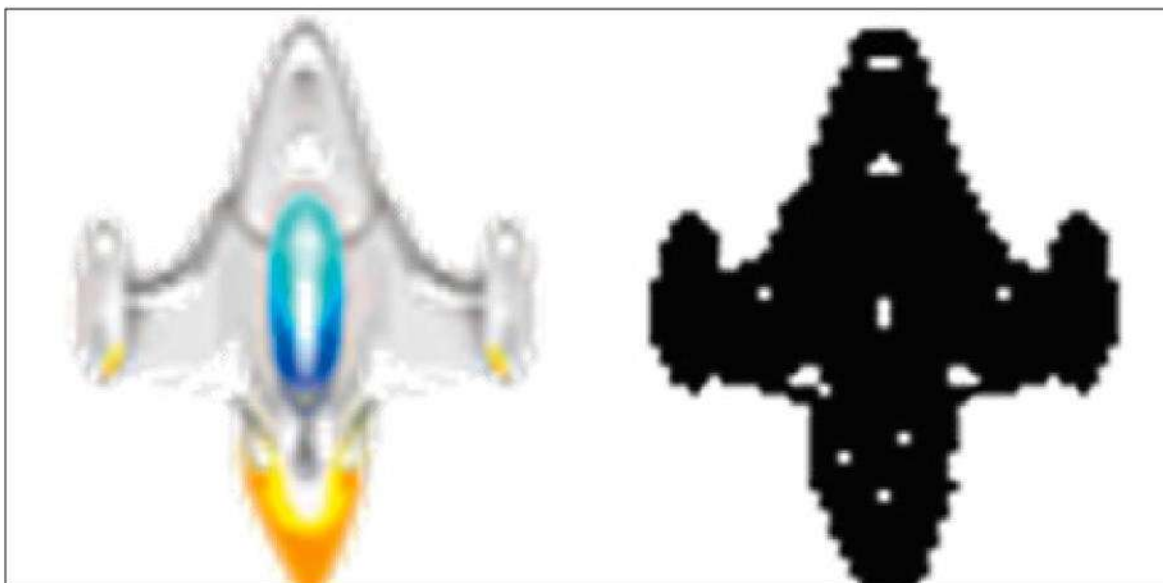
Get all the game code from David's Git at: <https://bit.ly/lxf314astro>





## QUICK TIP

Typedefs have been used to cut down on the number of \*s with pointers and to get rid of the struct keyword. For instance, a `firstpart` is the equivalent of `struct *firstpart`.



The player's ship on the left and its pixel mask on the right.

Every frame (one 50th of a second), we start by zeroing the cell's `numobject` in each of the 192 cells. As each object moves, the count in each cell it overlaps is incremented and a pointer to that object is stored in the cell. After all the objects have moved, we then look at all 192 cells and focus on those with a count of two or more. There's a further optimisation here. Instead of iterating through all 192 cells, we build up a list of cell indexes in the array `cellist` and increment an int counter called `numcells`. There might only be 10 possible collisions in a frame, so we can get the 10 cell indexes from this array and just process those.

## Collision or not?

There is no guarantee that a collision is happening; all that we know is that two or more objects have overlapped this cell. Also, if the count is more than two, then we need to check all of the possible collisions. Say there are three objects that overlap the same cell, we'll call them A, B and C. Then A might overlap B or C, and B might overlap C. So, for N objects, we have to check each possible pair. This is a mathematical combination and the number of checks is  $N!/(N-R)!R!$  where ! means a factorial.

For  $N = 3$ , that's  $3 \times 2 / 2 = 3$  checks, which are AB, AC and BC. For  $N = 4$ , that's  $4 \times 3 \times 2 / (2 \times 2) = 6$  checks. AB, AC, AD, BC, BD and CD.

## Bounding boxes

None of the object's shapes are exactly square but we do the initial check by assuming they are. A bounding box is the smallest rectangle that fits round an object; in this case, they're mostly squares. We'll use these bounding boxes to do the first check. If you run the game and press the Tab key, you'll see a red square drawn round each object. These show the bounding boxes as well as other information.

SDL2 has a function called `SDL_IntersectRect()`. It takes two `SDL_Rects` and returns a pointer to an `SDL_Rect` that has the coordinates of the overlapping area.

An `SDL_Rect` is just a very simple four int struct holding a point's coordinates and the width/height:

```
typedef struct SDL_Rect
```

```
{
    int x,y;
    int w,h;
} SDL_Rect;
```

The intersect rect where the two bounding boxes overlap is the key to detecting the possible collision. Although the bounding boxes overlap, the two objects clearly are not touching (see screenshot, page 94).

We need to check all the pixels in the intersect rect to see if any pixels from both objects occupy the same space. To do that, we need to compare the pixels in each object. We do that using pixel masks.

## What is a pixel mask?

The image on the left (above) is the 0-degree rotation of the player ship. All objects in the game that can rotate (player ship and all asteroids) come in 24 rotations of 15 degrees. Imagine if every colour pixel in this ship image was transformed into a single colour (value 1) and every other empty pixel was 0. That is a pixel mask and the image on the right represents that, with black for all of the 1-value pixels.

The files in the `masks` folders with the extension `.msk` are the pixel masks for all objects in the game, rotating or not. These are quite big files because they are one byte for every pixel. So, the 280x280 asteroids are 1,881,600 bytes in size. That's 280x280x24. This was a design decision, storing one pixel per byte. It could have been done with one bit per pixel, but then it would have needed 'and'ing to extract the bits, and this is one bit of code that needs to be fast. It's a classic memory versus execution time trade-off.

To see if there are any pixel collisions, the code iterates through every pixel in the intersect rect. We're not interested in the value of this pixel but instead we want its coordinate. We use that coordinate to look at the corresponding pixel in each of the two objects. It's quite a simple calculation. If object 1 is at (OX1, OY1) and the intersect pixel is at (IX1, IY1), we look up the pixel in the mask for OY1 at (IX1-OX1, IY1-OY1) and do the same for object 2, then compare the two pixels. It's slightly more complicated because we also need the rotation of each object, so we are looking up the

## QUICK TIP

Line 2199 has a commented out call to `ShowCells()`. If you uncomment this and recompile the game, when you run it you'll see the lines on the screen and the `numobjects` value showing how many objects are in each cell, as in the screenshot at the top of the previous page.



correct pixel mask. This is done in the function **Overlap** at line 1758.

### Game objects

All game objects are held in arrays. This means there's no need to allocate or free memory. However, to simplify the code, instead of indexing these objects, the individual element is assigned to a pointer. Look at **InitBullets()** on line 1,021. The **pbullet->active=0** is probably no faster than **bullets[i].active = 0**; if the compiler is optimising but it's slightly simpler to read. Also, it's easier to pass the pointer into a function:

```
for (int i = 0; i< MAXBULLETS; i++) {
    struct bullet *pbullet = &bullets[i];
    pbullet->active = 0;
    ...
}
```

If this wasn't done, the function **CheckAllObjects()** on line 1,841 would get very messy with **cells[x][y]** everywhere instead of **c**.

Because the pointers are just aliases for elements in an array, they can be passed around with impunity and don't need to be freed up. Some languages, such as C++, refer to these as weak references.

The objects in the game – player's ship, asteroids, bullets and alien ship(s) – are different types and sizes, and some rotate (player's ship, asteroids) and some don't. There's a struct called **firstpart** that is used in **Overlap()** for doing comparisons and adding objects to a cell. This is used for the asteroids, bullets, alien ship and player's ship:

```
struct firstpart {
    SDL_Rect r;
    int type;
    int active;
    int rotdir;
};
```

This means that you are not fighting the compiler when passing pointers to objects around. The **r** is the bounding box and x,y coordinate for objects. The **type** is a value 0-3 (defined as an enum but used as an int) for the object's type. The **active** field is a flag with 0 = dead, 1 = not dead, while **rotdir**

is the direction that it's facing, 0-23.

When an object hits another and gets destroyed in **DestroyObject()** at line 1700, we use a switch on the type, then cast it to the full size object in calls to **DestroyAsteroid()** and **DestroyAlienShip()**.

### Collision detection

If you look in **asteroids.c** line 15, you'll see a **#define SHOWOVERLAP** that is commented out. If you uncomment and recompile

the game, instead of blowing up the player's ship when it hits an asteroid or bullet, it carries on moving through the objects and shows every overlapping pixel in green.

Note: if you try that, you might also want to try commenting out lines 1,798 and 1,800, and in lines 1,056 and 2,298 set the **GameLevel** to 49 and then recompile, so you get lots of big asteroids. This also lets asteroids move through each other without blowing up and you get some very large green

asteroids as they pass. It's quite spectacular but can drop the frame rate down to as low as four frames per second. That's not surprising when you get two 280x280 pixels asteroids overlapping. The intersect rect will have tens of thousands of bytes in it and that's a lot of lookups.

Without the **SHOWOVERLAP** **#define**, objects are destroyed as soon as two pixels touch.

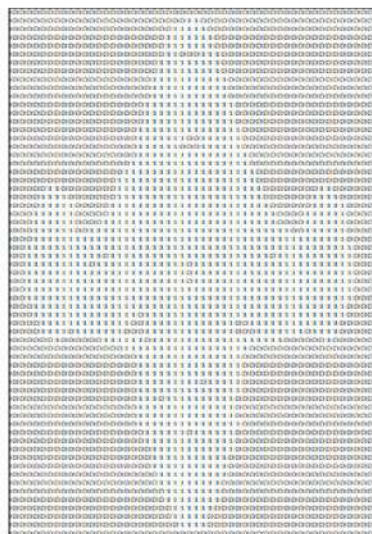
Using 64x64 as the cell size might not be the optimum size. A bigger cell would mean a screen had fewer cells and that might speed up comparisons. However, this might mean more objects could overlap a cell. For five objects, it needs 10 comparisons, while for six it needs 15. **LXF**



Asteroids compiled with the **SHOWOVERLAP** **#define** – this shows the player's ship moving through an asteroid.

### QUICK TIP

The SDL2 library has an internal constant for the maximum width of texture it can load (x86/64 Linux, Windows and Raspberry Pi are all different). The full width of 6720 (280x24 images) was too big for one, so the four asteroid images and the player's ship images were each split over two rows.



Mask of player's ship image 0 generated by the Genmasks utility.

## » HOW TO CREATE PIXEL MASKS

This is an easy task to automate. With the game files is **genmasks.c**, which does the job. The SDL library lets you use surfaces and textures for images. Surfaces are a data structure in normal RAM while textures are similar but in video RAM. To compile this, take **tasks.json** and remove the line with **levels.c**.

The game uses textures for speed, but for the PNG-to-mask conversion,

surfaces work really well. The image is loaded into a surface and then directly read pixel by pixel into a buffer converting colour pixels to 1 and background pixels to 0. The buffer is then written as a binary file. There's a function called **TestMask** that reads a specified mask and outputs it as a text file like this, made up of ones and zeros, to validate the mask. Viewing

it in a text editor makes it look a little stretched.

You only need to use this utility if you are creating your own graphics. If you add a graphic, adjust **NUMsurfaces** and put the path in the **surfacenames** array, then add a call to **ProcessSurface()** in the **ProcessSurfaces()** function. Note this uses a texture to display the image on screen during mask creation.

» **IMPROVE YOUR CODE SKILLS** Subscribe now at <http://bit.ly/LinuxFormat>

NEXT MONTH

**LXF315**  
will be on sale  
Tuesday 30th  
April 2024

# 20 YEARS OF UBUNTU

We dive into the brand-new Ubuntu 24.04 LTS and look back at where it all began and how we got here!

## Linux desktop basics

Get up to speed using the Gnome desktop. We guide you through the latest release and how to get more from it.

## Thunderbird on fire

Live your best email life! We're taking *Betterbird* for a spin, the remix of the classic email client you need to use.

## The ultimate hacker!

We try out the finest hacking distros on the market, so you can defend and test your systems with the best.

## The 2038 apocalypse cometh

Well, no, as with Y2K, the boffins are on top of the problem, but we're looking at the Unix epoch for giggles.

Content of future issues subject to change. Twenty years?! We feel so old...

CREDIT: Magictorch

# LINUX

The #1 open source mag

# FORMAT

Future Publishing Limited, Quay House, The Ambury, Bath, BA11UA  
Email [contact@linuxformat.com](mailto:contact@linuxformat.com)

### EDITORIAL

Editor-in-chief Neil Mohr

Art editor Fraser McDermott

Production editor Katharine Davies

Group editor-in-chief Graham Barlow

Group art director Warren Brown

### Editorial contributors

Mats Tage Axelsson, Mike Bedford, Jonni Bidwell, David Bolton, Neil Bothwick, Stuart Burns, Barry Collins, Nate Drake, Kseniya Fedoruk, Tam Hanna, Matt Holder, Jon Masters, Nick Peers, Les Pounder, Michael Reed, Mayank Sharma, Shashank Sharma, Andrew Sink, Chris Szweczyk

Cover illustration Magictorch.com

Microsoft Windows is a trademark of the Microsoft group of companies.

ChatGPT and the GPT logo are property of OpenAI.

Raspberry Pi is a trademark of the Raspberry Pi Foundation.

Tux credit: Larry Ewing ([lewing@isc.tamu.edu](mailto:lewing@isc.tamu.edu)).

Content production Adequate Media Limited

### ADVERTISING

Commercial sales director Clare Dove

[clare.dove@futurenet.com](mailto:clare.dove@futurenet.com)

Advertising director Lara Jaggon

[lara.jaggon@futurenet.com](mailto:lara.jaggon@futurenet.com)

Account director Andrew Tilbury

[andrew.tilbury@futurenet.com](mailto:andrew.tilbury@futurenet.com)

### INTERNATIONAL LICENSING

Head of print licensing Rachel Shaw

Linux Format is available for licensing and syndication.

To find our more contact us at [licensing@futurenet.com](mailto:licensing@futurenet.com)

or view our content at [www.futurecontenthub.com](http://www.futurecontenthub.com)

### NEW SUBSCRIPTIONS & BACK ISSUES

Web [www.magazinesdirect.com](http://www.magazinesdirect.com)

UK 0330 333 1113 World +44 (0) 330 333 1113

### EXISTING SUBSCRIPTIONS

Web [www.mymagazine.co.uk](http://www.mymagazine.co.uk)

UK 0330 333 4333 World +44 (0) 330 333 4333

Subscription delays: Disruption remains within UK and international delivery networks. Please allow up to seven days before contacting us about a late delivery to [help@magazinesdirect.com](mailto:help@magazinesdirect.com)

### CIRCULATION

Newstrade & retail category director Ben Oakden

### PRODUCTION AND DISTRIBUTION

Group head of production Mark Constance

Production manager Nola Cokely

Senior ad production manager Jo Crosby

Digital editions manager Jason Hudson

### THE MANAGEMENT

Managing director technology group Paul Newman

Global head of design Rodney Dive

Commercial finance director Tania Brunning

Printed by William Gibbons & Sons

Distributed by Marketforce UK

121-141 Westbourne Terrace, London W2 6JR. [www.marketforce.co.uk](http://www.marketforce.co.uk)

For enquiries email: [mfcommunications@futurenet.com](mailto:mfcommunications@futurenet.com)

**Order and access back issues:** If you are an active subscriber, you have instant access to back issues through your iOS or Android devices. Your digital magazine entitlement is available at no additional cost and no further action is required. Pocketmags library may not have access to the full archive of digital back issues. You will only be able to access the digital back issues as long as you are an active subscriber.

**To purchase single back issues (print format only):** Visit [www.magazinesdirect.com](http://www.magazinesdirect.com) (click on Single Issues tab) or email [help@magazinesdirect.com](mailto:help@magazinesdirect.com). For further help call +44(0)330 333 1113. Lines are open Mon-Fri 8.30am-7pm and Sat 10am-3pm UK time. Magazinesdirect.com is owned and operated by Future Publishing Limited.

**BAR rates for Linux Format:** £84.37 for UK, £171 for Europe, \$194 for USA, £149 for rest of world.

Linux® is the registered trademark of Linus Torvalds in the US and other countries. GNU/Linux is abbreviated to Linux throughout for brevity. Where applicable, code printed in this magazine is licensed under the GNU GPL v2 or later. See [www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html). All copyrights and trademarks are recognised and respected.

**Disclaimer:** All contents © 2024 Future Publishing Limited or published under licence. All rights reserved. No part of this magazine may be used, stored, transmitted or reproduced in any way without the prior written permission of the publisher. Future Publishing Limited (company number 2008885) is registered in England and Wales. Registered office: Quay House, The Ambury, Bath BA11UA. All information contained in this publication is for information only and is, as far as we are aware, correct at the time of going to press. Future Publishing Limited cannot accept any responsibility for errors or inaccuracies in such information. You are advised to contact manufacturers and retailers directly with regard to the price of products/services referred to in this publication. Apps and websites mentioned in this publication are not under our control. We are not responsible for their contents or any other changes or updates to them. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.

If you submit material to us, you warrant that you own the material and/or have the necessary rights/permissions to supply the material and you automatically grant Future Publishing Limited and its licensees a licence to publish your submission in whole or in part in any/all issues and/or editions of publications, in any format published worldwide and on associated websites, social media channels and associated products. Any material you submit is sent at your own risk and, although every care is taken, neither Future nor its employees, agents, subcontractors or licensees shall be liable for loss or damage. We assume all unsolicited material is for publication unless otherwise stated, and reserve the right to edit, amend and adapt all submissions. All contents in this magazine are used at your own risk. We accept no liability for any loss of data or damage to your systems, peripherals or software through the use of any guide.

We are committed to only using magazine paper derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this magazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards.

Future is an award-winning international media group and leading digital business. We reach more than 57 million international consumers a month and create world-class content and advertising solutions for passionate consumers online, on tablet & smartphone and in print.



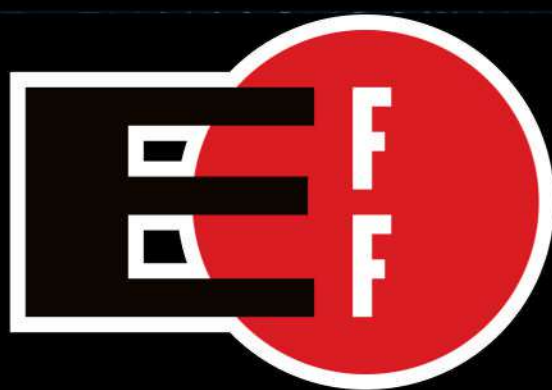
Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR) [www.futureplc.com](http://www.futureplc.com)

Chief Executive Officer **Jon Steinberg**  
Non-Executive Chairman **Richard Huntington**  
Chief Financial and Strategy Officer **Penny Ladin-Brand**

Tel +44 (0)1225 442 244







The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

**EFF.ORG**

**ELECTRONIC FRONTIER FOUNDATION**

Protecting Rights and Promoting Freedom on the Electronic Frontier



**THE  
BRAIN  
TUMOUR  
CHARITY**

**A CURE CAN'T WAIT**

**BRAIN TUMOURS  
MOVE FAST.  
WITH YOUR HELP,  
WE CAN TOO!**

We're working to create a future where brain tumours are curable.  
We urgently need your help to accelerate research.

Text DEFEAT5 to 70507 to donate £5, please help us to find a cure.

**[thebraintumourcharity.org](https://thebraintumourcharity.org)**

© The Brain Tumour Charity 2020. Registered Charity in England and Wales  
(1150054) and Scotland (SC045081)



Registered with  
**FUNDRAISING  
REGULATOR**